University of Cambridge

Department of Engineering

Multiresolution surfaces in shape optimisation of shells and solids

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

by

Konara Mudiyanselage Kosala Bandara

Churchill College

December 2013

Acknowledgements

First and foremost I would like to express my sincere gratitude to my supervisor Dr Fehmi Cirak. Not only did he introduce me to this fascinating research field, but also was the main source of motivation and guidance throughout my study. His patience and dedication is greatly appreciated and I will forever treasure the time spend at the Computational Structural Mechanics Lab under his guidance.

My advisor, Dr Matthew DeJong provided valuable feedback on my research progress which is greatly appreciated.

I would like to thank Dr. Thomas Rüberg and Dr. Burkhard Bornemann for their contribution to openFTL software, in which the research presented in this dissertation is implemented. Additionally, the technical support given by them during implementation is much appreciated.

I would like to express my gratitude to all former and current members of my research group for their friendship: Matija Kecman, Dr. Quan Long, Musabbir Majeed, Ray Rui, Richard Silveira, Dr. Jakub Sistek and others.

I am thankful for the financial support from the Cambridge Commonwealth trust and Churchill college which enabled me to pursue my studies at Cambridge University.

Finally, I would like thank to my wife Samila, and my family for their boundless encouragement, understanding and support throughout this study.

Declaration

I wish to declare that, except for commonly understood ideas and concepts, or where specific reference is made to the work of other authors, the contents of this dissertation are original and do not include subject matter that is the outcome of work done in collaboration with others. The research described in this dissertation was conducted in the Department of Engineering at the University of Cambridge from October 2009 to September 2013. This dissertation has not previously been submitted, in part or in whole, to any other university or institution for any degree, diploma, or other qualification. This dissertation is presented in 164 pages, and contains 97 figures and approximately 34,000 words, including references and appendices.

Konara Mudiyanselage Kosala Bandara

Abstract

Traditionally, geometric design, analysis and optimisation, the main components of a product development cycle, are treated as separate modules requiring different methods and representations. Recent progress towards addressing this issue has been focused on Isogeometric analysis where the same basis functions are used for both geometry and analysis representations. However, having the same mesh for both design and analysis causes problems in the optimisation module as a result of having too many design variables. Traditional parameter based shape optimisation sidesteps this problem by restricting the space of possible geometry changes during optimisation.

In a novel approach, this dissertation proposes the use of multiresolution surfaces for shape optimisation of shells and solids. A hierarchy of meshes are created with the coarse resolutions used in the geometric design and optimisation modules whereas the fine resolution is used in the analysis module. In addition to facilitating parameter-free optimisation, enabling more flexibility in design space exploration, the proposed method is capable of using different solver modules including non-isogeometric solvers. Two alternative concepts, one based on subdivision surfaces and the other using progressive meshes, are explored for creating the multiresolution representation.

Kirchhoff-Love shells offer an ideal formulation for optimisation of shells due to the displacement based setting without any rotational degrees of freedom. The proposed multiresolution framework is demonstrated for shape optimisation of Kirchhoff-Love shells using analytically derived discrete shape derivatives.

Use of conventional finite element methods for analysis of solid geometry require a volumetric mesh which is likely to be distorted during shape optimisation leading to inaccuracies. Two possible solutions to this problem are offered by immersed and boundary element methods that only require a surface mesh representing the domain boundary to be deformed. The former is used in the thesis for shape optimisation of linear elastic solids. The developed top-down multiresolution method is also demonstrated for shape optimisation in electrostatic problems using a boundary element solver.

Numerous validation and industry strength examples are presented to highlight the merits of using the proposed multiresolution shape optimisation paradigm for integrated product development involving shells and solids.

Contents

List of Figures

1. Introduction

1.1. Motivation

Computer aided design (CAD) and finite element analysis (FEA) are integral components in product development today. CAD is used to build a precise geometric descriptions of the design model while FEA simulations allow increased performance without time consuming physical testing. Simulation driven product development, where the geometric design is iteratively optimised based on simulation feedback, is the main focus of this dissertation. The key requirement for such a framework is the seamless exchange of information between the main modules, i.e. design, analysis and optimisation.

Integrated design analysis and optimisation

The incompatibility between the geometry representations has been a major bottleneck in integrating CAD and FEA. Historically, the finite element method widely used for simulation was formulated much earlier than the geometry descriptions used in CAD were developed. The former favours body-fitted polygon meshes whereas B-spline based Nonuniform Rational B-splines (NURBS) or subdivision methods are typically used in the latter. Presently, this implies a separate representation, i.e. a finite element mesh, has to be maintained that approximates the design CAD geometry. This requires regenerating the FEA mesh via various meshing tools each time the design is changed, as a direct link between the two geometries cannot be maintained (Figure 1.1). In high-end production industries such as automotive, aerospace, it is hardly surprising that some estimates put the amount of time spent on mesh generation at 80% of total analysis time [73]. Figure 1.2 shows the three different representations using an automotive model with distinct design features. Note the smooth NURBS and subdivision surfaces in comparison to the fine polygon mesh used for FEA.

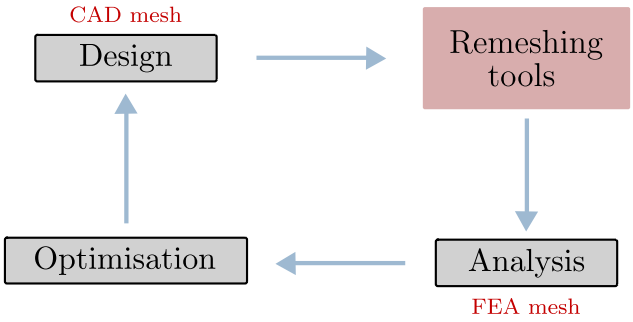
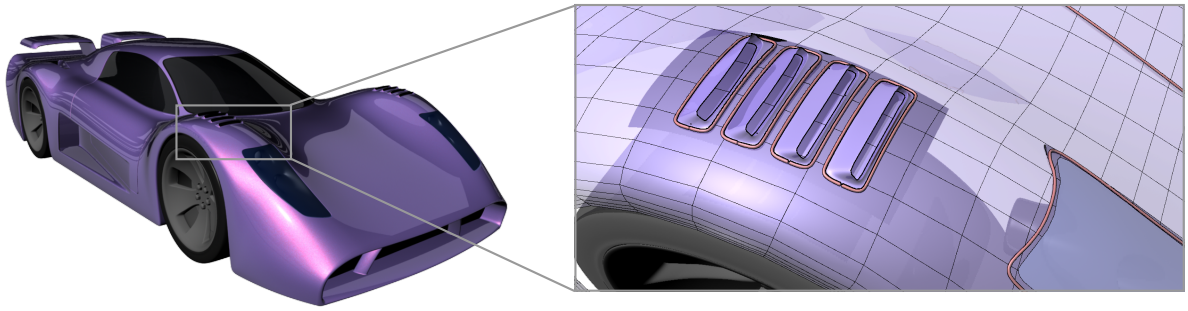


Figure 1.1.: Traditional product development framework. Geometry is remeshed each time the design is changed.

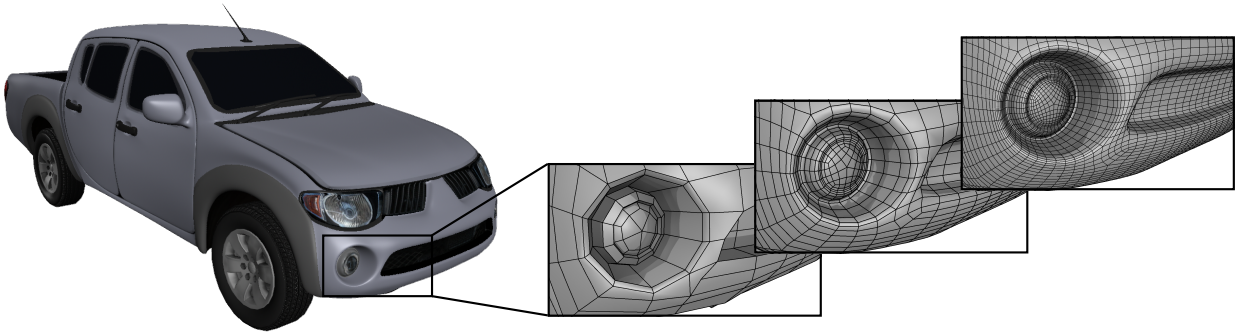
Isogeometry paradigm

In recent times the major drive towards integrating CAD and FEA has come from isogeometric analysis where the same representation is used for both geometric design and analysis [33, 73]. The key innovation is to use either subdivision or NURBS as the basis functions in finite element analysis. However presence of multiple NURBS patches and trimming curves, as often is the case in real-world NURBS applications, causes deficiencies in NURBS based isogeometric methods. T-splines [116] has emerged as a candidate to circumvent this issue, in which T junctions are used to create a single watertight NURBS model. However isogeometric methods based on T-splines [9] suffer from linear dependency problems [19].

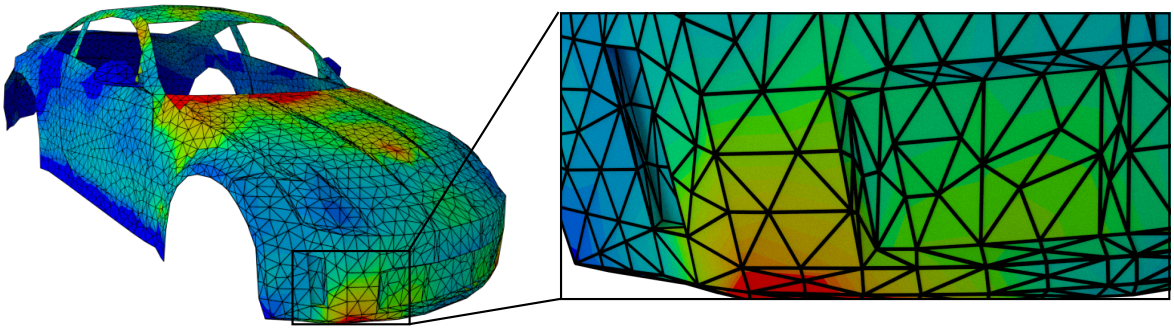
Although direct transfer of data from CAD to FEA is possible with isogeometric methods, using the same geometric mesh for design and analysis causes many problems in the optimisation module. Essentially for most practical design optimisation problems, having too many design variables will result in the optimal solutions being wiggly irregular shapes that cannot be used for realistic designs [18].



(a) NURBS geometry. Non-watertight control mesh with trimming curves (in red) used to show/hide wanted/unwanted regions.



(b) Subdivision geometry. Repeated refinement of a watertight coarse control mesh creates a smooth surface in the limit.



(c) Typical body fitted polygon mesh used in FEA. Requires removal of gaps, overlaps when created from a NURBS model.

Figure 1.2.: Geometry descriptions used in CAD and FEA. Geometric models obtained from http://www.turbosquid.com and http://www.grabcad.com .

1.2. Methodology

Shape optimisation

Shape optimisation seeks an optimal geometric shape among alternative designs by minimising a performance related objective function subject to constraints.

$$\text{minimise} \qquad \mathcal{J}(\boldsymbol{s}, \boldsymbol{u}(\boldsymbol{s})) \quad \boldsymbol{s} \in \mathbb{R}^{n_s} \tag{1.1a}$$

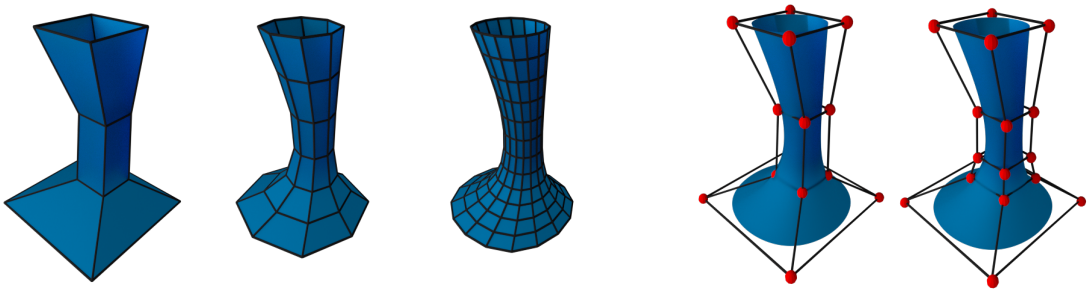
$$\text{such that} \quad g_i(\boldsymbol{s}, \boldsymbol{u}(\boldsymbol{s})) = 0 \quad i = 1, \ldots, n_g \tag{1.1b}$$

where $\boldsymbol{s}$ is the vector of design variables related to the geometric shape of the domain and $\boldsymbol{u}$ is the vector of state variables, i.e. displacement, that depend on $\boldsymbol{s}$. Additional constraints, i.e. equilibrium, are denoted by the set g_i. For simplicity, only equality constraints are assumed here. Mathematical programming methods used to minimise (1.1) can be either gradient based or non-gradient based. The former is usually employed as it utilises more information in the form of design sensitivity

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{s}}(\boldsymbol{s}, \boldsymbol{u}(\boldsymbol{s})) = \frac{\partial \mathcal{J}}{\partial \boldsymbol{s}} + \frac{\partial \mathcal{J}}{\partial \boldsymbol{u}} \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{s}} \tag{1.2}$$

which is the derivative of the objective function with respect to the design variables $\boldsymbol{s}$. The natural choice for design variables is to select the nodal coordinates of the finite element mesh itself leading to parameter-free optimisation. However having such a one-to-one correspondence between the design variables and the finite element mesh can quickly lead to unrealistic designs and is generally avoided. Informally in such situations the non-convexity of the optimisation problem can easily lead to optimal designs belonging to a local minima. Examples of this nature are provided in later chapters. Historically, the issue was side-stepped using various methods that restrict possible boundary deformations by resorting to a reduced number of auxiliary parameters [63]. This practice is known as parameter optimisation and is widely used in industrial product development today.

It is clear that parameter-free optimisation enables more flexibility in design space exploration and obtaining innovative designs. Additionally, advances in additive manufacturing imply that manufacturability of complex geometries is no longer a constraint. Isogeometric analysis is a potential candidate for parameter-free shape optimisation since the B-spline basis functions can be used to maintain the desired regularity of the shape.



(a) Subdivision refinement. From left to right; coarse control mesh, once refined mesh, twice refined mesh.

(b) Knot insertion in NURBS. Note that only four knots are inserted here.

Figure 1.3.: Mesh refinement in NURBS and subdivision.

In this setting, the control points can have the combined role of being the design variables during optimisation and the degrees of freedom during analysis. This is demonstrated by, amongst others, Bletzinger and Ramm [16] using Bézier patches, Cirak et al. [30] using subdivision, Wall et al. [128] with NURBS and Seo et al. [117], Ha et al. [60] using T-splines.

The key difference between isogeometric optimisation methods and the present work is the use of multiresolution geometry editing techniques originating from computer graphics. It will be shown how the proposed method avoids many of the pitfalls of parameter-free optimisation. Additionally, the method is specially beneficial in optimisation of solids where any shape changes distort the internal mesh. In the present work, this issue is avoided by using immersed and boundary element methods that only require a surface mesh representing the domain boundary to be deformed. In this context, the multiresolution hierarchy is constructed for the boundary surface mesh typically given as a b-rep (boundary representation) in a CAD model.

Multiresolution framework

Subdivision surfaces have been widely used in the animation industry due to their ability to represent arbitrary geometry with smooth watertight meshes [41]. In a subdivision setting, repeated refinement of the control mesh creates a hierarchy of higher resolution meshes that ultimately converge to a limit surface as shown in Figure 1.3a. Refinement of the control mesh is achieved via a *subdivision matrix* $\mathbf{S}$, which relates the control points of resolutions ℓ and $\ell+1$. Given a vector of control points $\boldsymbol{x}^\ell$ in resolution ℓ, the

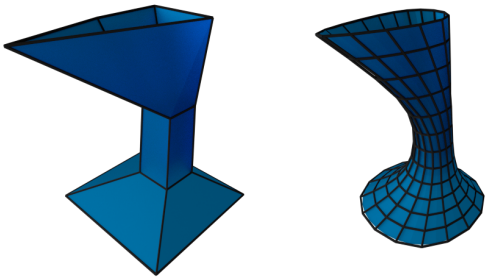


Figure 1.4.: Subdivision editing. A single control point from the coarse control mesh in Figure 1.3a is perturbed (left) resulting in a smooth shape change in the twice refined mesh (right).

fine resolution control points $\boldsymbol{x}^{\ell+1}$ can be obtained by multiplication with a subdivision matrix

$$\mathbf{x}^{\ell+1} = \mathbf{S}\mathbf{x}^{\ell} \tag{1.3}$$

The equivalent process to increasing mesh resolution in NURBS is referred to as knot insertion (Figure 1.3b). Observe that the fine resolution subdivision mesh resembles a conventional body-fitted finite element mesh, which can be altered in a smooth, regulated manner by moving the control points of the coarse control mesh (Figure 1.4).

The methodology proposed in this dissertation for creating an integrated design, analysis and optimisation framework using parameter-free shape optimisation is mainly motivated by the following two properties of subdivision;

- *Multiresolution property*: Subdivision provides a natural means of navigating between different mesh resolutions. A coarse resolution model can be used as the CAD model and corresponding fine resolution model as the analysis model with subdivision linking the two.

- *Variation diminishing property*: This property, inherited from B-splines, enables changes to the coarse design model to be propagated to the fine resolution analysis model in a smooth and robust manner.

The conceptual multiresolution framework proposed is shown in Figure 1.5. A hierarchy of meshes are created with the coarse resolutions used in design and optimisation modules whereas a fine resolution is used in the analysis module. The modular nature of the whole setup implies that the multiresolution geometric representation can be independent of the analysis module. Hence the strict requirement of an isogeometric solver is not required

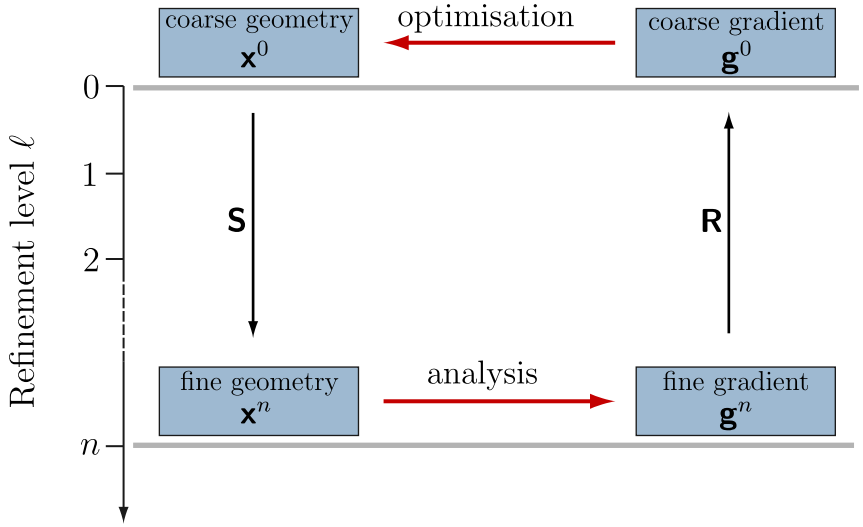


Figure 1.5.: Subdivision based multiresolution framework. Two models of different resolution are maintained; a coarse resolution model with the CAD design variables and a fine resolution FEA model. Information is exchanged between the two resolutions using (1.3) and (1.4). Note that only a simplified presentation is made here, the intermediate resolutions are hidden for brevity.

and the simulation module can use any type of solver capable of using conventional body-fitted finite element meshes. The design sensitivities (1.2) are computed with the fine resolution analysis mesh and need to be projected to the coarse resolution containing the actual design variables. A wavelet inspired subdivision coarsening method is developed to this end. This coarsening operator functions in an opposite manner to the subdivision matrix (1.3) and uses the control points of resolution $\ell + 1$ to yield those of resolution ℓ

$$\mathbf{x}^{\ell} = \mathbf{R}\mathbf{x}^{\ell+1} \tag{1.4}$$

Clearly any B-spline based geometry description such as NURBS, subdivision, T-splines among others can be used to create a multiresolution framework due to the refinability property of B-splines. Such a framework can be used to integrate design, analysis and optimisation to create an iterative workflow for evolving designs from a concept stage to final production level (Figure 1.6). The limitation is that the initial input geometry is required to be representable using a coarse control mesh.

However real world design examples often contain many complex geometric features that cannot be resolved using a coarse mesh. In an alternative point of view, a multiresolution mesh hierarchy can be created from such a highly detailed fine resolution geometry, with an opposite starting point from the previously described subdivision based framework.

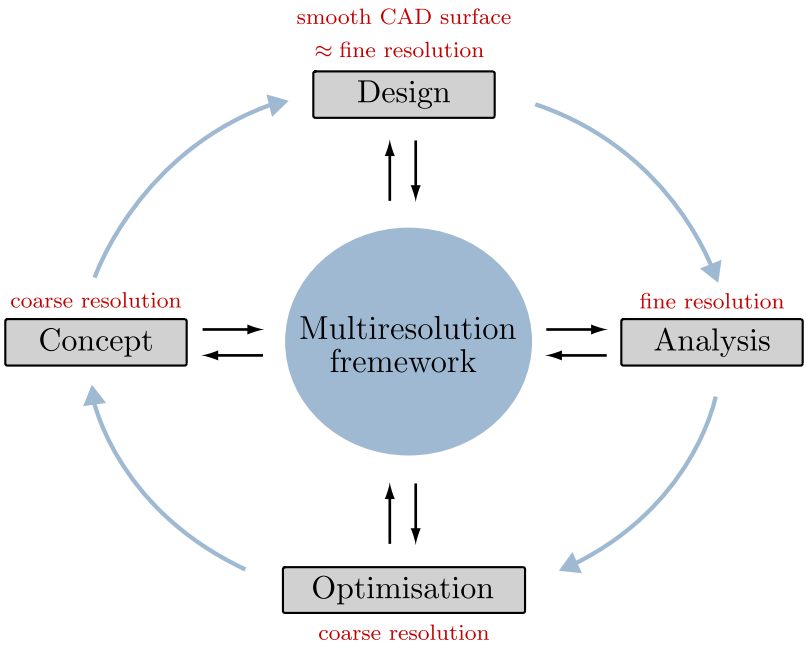


Figure 1.6.: Isogemetric product development cycle based on a multiresolution framework. Concept stage creates a coarse control mesh on which the design variables are defined. This has a corresponding smooth CAD model which represents the current design. The analysis stage uses a suitable fine resolution mesh that approximates the CAD model.

A family of tools based on *mesh decimation* can be used to this end, an example of which is shown in Figure 1.7. Mesh decimation, also known as mesh simplification, is aimed at reducing the number of nodes (vertices) in a geometric mesh and can be achieved in a number of ways such as edge collapse or facet merge [67]. The decimation process can be expressed similar to the subdivision coarsening process (1.4) with

$$\mathbf{x}^{\ell-1} = \mathcal{R}(\mathbf{x}^{\ell}) \tag{1.5}$$

Observe in Figure 1.7 that geometric features are lost during mesh decimation. Such missing features, called *details* hereafter, need to be recorded if a loss-less mapping between coarse and fine resolutions is to be achieved. As will become clear, this is vital for translating coarse resolution design changes to the original fine resolution model. The present work explores multiresolution geometry editing methods based on progressive mesh simplification [69] for multiresolution editing (Figure 1.8). The method is based on progressively storing details for each decimation step. The details $\mathbf{d}^{\ell}$ representing the

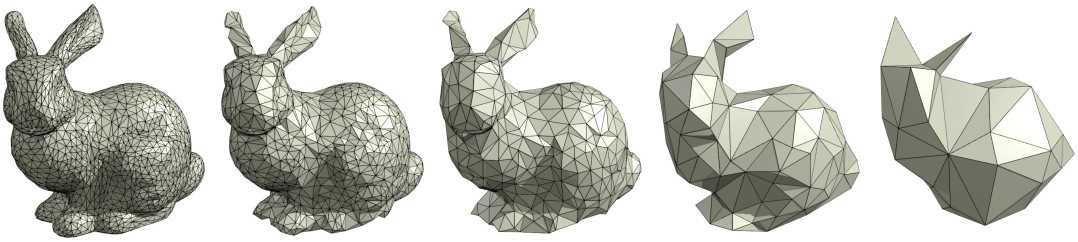


Figure 1.7.: Mesh decimation of the Stanford bunny. Percentage of vertices removed from left to right: 0%, 50%, 80%, 95%, 98%.

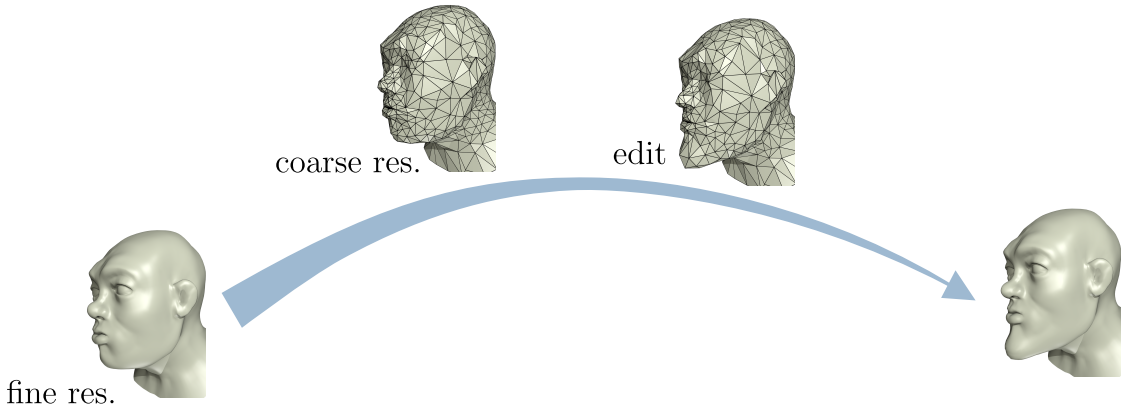


Figure 1.8.: Multiresolution editing using progressive meshes. An initial fine resolution model is coarsened and edited. Next the changes due to the coarse resolution edit are translated back to the fine resolution.

difference between mesh resolutions ℓ and $\ell - 1$ can be computed as follows

$$\mathbf{d}^{\ell} = \mathbf{x}^{\ell} - \mathcal{S}(\mathbf{x}^{\ell-1}) \tag{1.6}$$

The presence operator $\mathcal{S}$, which is essentially a smoothing operator similar to subdivision, helps in preserving smoothness of the geometry during multiresolution editing. Suitable subdivision schemes for this are explored in later chapters. Assume the coarse resolution control points are now edited during optimisation, $\mathbf{x}^{\ell-1} \to \tilde{\mathbf{x}}^{\ell-1}$. The changes can be transferred to the fine resolution, termed *detail restoration*, as follows

$$\tilde{\mathbf{x}}^{\ell} = \mathcal{S}(\tilde{\mathbf{x}}^{\ell-1}) + \mathbf{d}^{\ell} \tag{1.7}$$

Figure 1.9 shows a simplified multiresolution framework based on progressive meshes. At the start of the cycle, a given fine resolution mesh is decomposed to create the details and a coarse resolution mesh using (1.6). After this initial step, the optimisation workflow

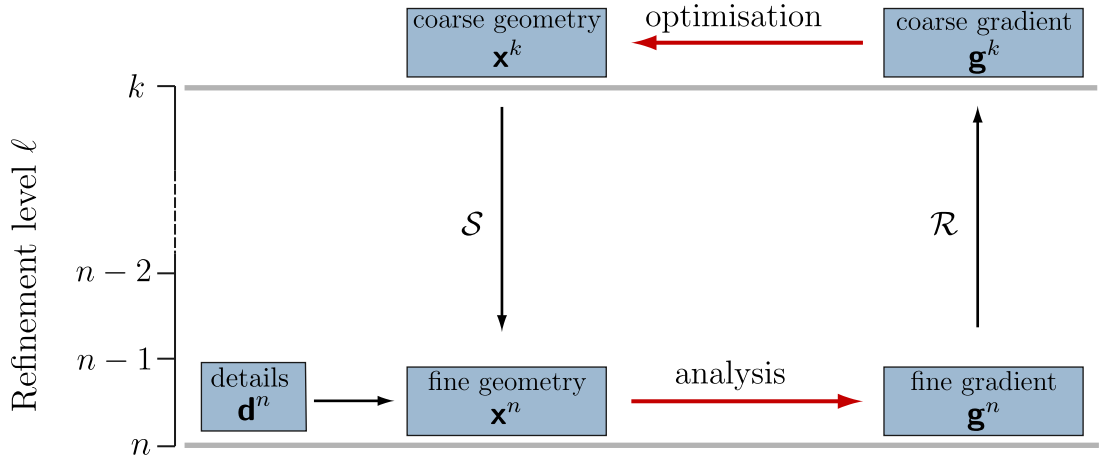


Figure 1.9.: Progressive mesh based multiresolution framework. The original fine resolution CAD model is decomposed into a coarse resolution model containing the design variables and a set of details related to the difference of the two. The FEA model is the same as the original fine resolution CAD model. Information is exchanged between the two resolutions using (1.5) and 1.7. Note that only a simplified presentation is made here, the intermediate resolutions and details are hidden for brevity.

is very similar to the subdivision based framework.

In summary, two multiresolution frameworks based on subdivision (Figure 1.5) and progressive meshes (Figure 1.9) are proposed in this dissertation. Both approaches are fundamentally based on the same concept of using a multiresolution hierarchy to connect a fine resolution analysis model to a coarse resolution model containing the design variables. This creates an inherently integrated design, analysis and optimisation framework (Figure 1.6). More importantly, maintaining the design variables in a coarse resolution eliminates the shortcoming of parameter-free optimisation. Different stages of product development involve varying degrees of emphasis on globalisation of geometric design features. In the initial design stage, i.e. concept development stage, global shape changes are expected whereas shape changes during later design stages tend to be more minimal and localised. This is mirrored in the proposed setup where, during the concept stage the design variables can be control points of a very coarse control mesh but transferred to finer resolutions in later design stages without any interference to the analysis model.

1.3. Objectives and layout

The main objective of this dissertation is to explore the use of mutiresolution surfaces for integrated design, analysis and optimisation of shells and solids. The first task is the realisation of a multiresolution geometric framework capable of translating design and analysis information between the different modules involved. Two alternatives exist as previously discussed; a top-down approach starting from a design that can be represented using a coarse resolution or a bottom-up approach starting from a design already containing many design features.

Chapters 2 and 3 review the definition of NURBS, subdivision and progressive meshes as potential candidates for realising the desired multiresolution framework. Specifically, Chapter 2 compares the merits of using either a NURBS or a subdivision representation for the top-down approach. Progressive meshes are reviewed in Chapter 3 to facilitate the opposite bottom-up framework.

In Chapter 4, the integrated design, analysis and optimisation cycle is formulated for subdivision and progressive mesh representations. Chapter 5 demonstrates the proposed framework for shape optimisation of thin shells. Discrete sensitivities are derived analytically for Kirchhoff-Love shells and computed with subdivision shells [33]. Chapter 6 explores the optimisation of solid bodies using immersed and boundary element methods for the solver module. The immersed method from Rüberg and Cirak [110] is used for shape and topology optimisation of linear elastic solid bodies. The extension to topology optimisation is made via the bubble method [46]. Additionally, the shape optimisation of solid bodies in electrostatics is presented using a boundary element method [102] to solve the Laplace equation. Finally conclusions are drawn and future research directions explored in Chapter 7.

2. CAD geometry representation techniques

This chapter reviews the definition of NURBS and subdivision surfaces starting from their common foundation of B-splines. See Böhm [17] for an overview of different CAD representation techniques in a historical perspective. Subsequently, the possibility of using each geometry description for the desired multiresolution framework is examined.

2.1. B-splines

The name spline is derived from the thin strip of wood held in place by weights used by early draftsmen to generate smooth curves. B-splines or basis splines originate from Bézier curves developed in the 1960s, which represented polynomial curves using Bernstein polynomials. In contrast to Bézier curves which are comprised of a single polynomial segment, B-splines are piecewise polynomial curves with multiple polynomial segments. The continuity of the segments is defined by a *knot vector* $\boldsymbol{u}$ which contains the parameter values u_i, known as *knots*, that map into the ends of each polynomial segment. The definition of a B-spline curve is as follows;

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{x}_i \qquad \boldsymbol{u} = \{u_0, u_1, \ldots, u_{n+p+1}\} \tag{2.1}$$

where $N_{i,p}$ are polynomial *basis functions* of degree p and the geometric coefficients $\boldsymbol{x}_i$ are the *control points*. Figure 2.1 shows an example of a cubic B-spline curve and the corresponding B-spline basis functions.

B-splines have several unique properties that make them ideal for CAD geometry. A few such properties that are relevant to the present work are explained below. See Prautzsch

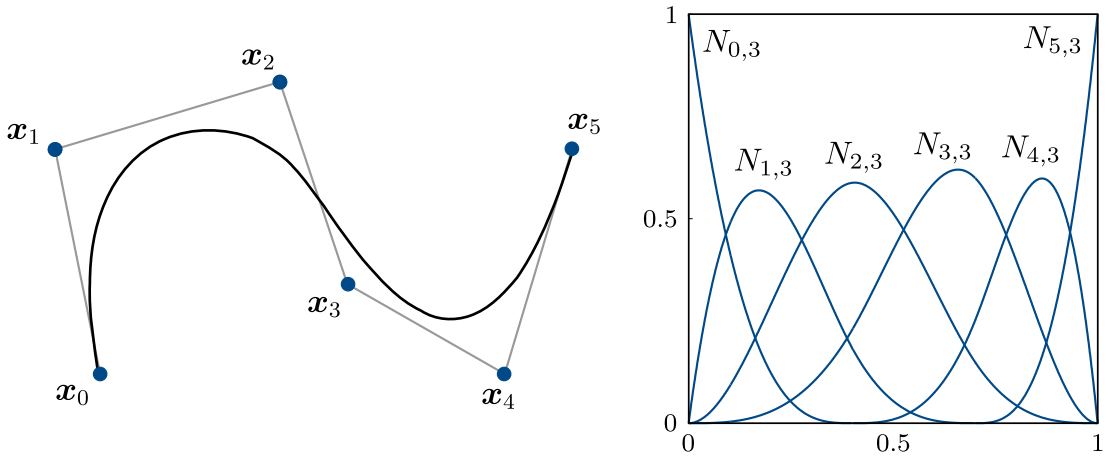


Figure 2.1.: Cubic B-spline curve and basis functions. It is important to distinguish the difference between the *control polygon* connecting the control points in blue and *limit curve* in black. The knot vector $\boldsymbol{u} = \{0, 0, 0, 0, 0.4, 0.7, 1, 1, 1, 1\}$.

et al. [107] and Piegl and Tiller [105] for more details.

- *Variation diminishing property*: No straight line intersects the curve more times than it does the control polygon. Essentially this means the curve cannot be more 'wiggly' than the control polygon.

- *Local support*: The basis functions have a local support such that $N_{i,p} \neq 0$ only if u is inside the knot interval $[u_i, u_{i+p+1}]$. This implies that each basis function influences only a limited region of the overall geometry, a property distinct from Bézier curves.

- *Recurrence formula*: A recurrence relation can be used to compute successively higher degree B-spline basis functions. The zero degree B-spline is the box function

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

  Higher degree B-splines are computed with the recurrence relationship

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.3}$$

- *Refinability*: The B-spline basis functions can be expressed as translated and dilated copies of themselves. For uniform B-splines, i.e. the distance between the

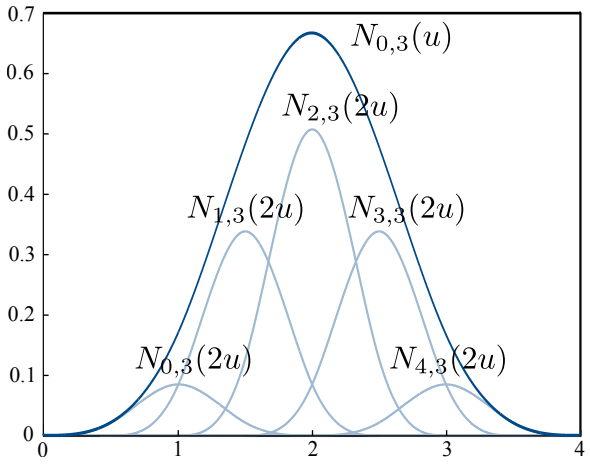


Figure 2.2.: Refinement of uniform Cubic B-spline basis function (2.4).
$N_{0,3}(u) = \frac{1}{8}N_{0,3}(2u) + \frac{1}{2}N_{1,3}(2u) + \frac{3}{4}N_{2,3}(2u) + \frac{1}{2}N_{3,3}(2u) + \frac{1}{8}N_{4,3}(2u)$

knots are uniform, the following two scale relation holds (see Figure 2.2)

$$N_{i,p}(u) = \frac{1}{2^p}\sum_{j=0}^{p+1}\binom{p+1}{j}N_{i,p}(2u-j) \tag{2.4}$$

where the refined basis function are multiplied with the binomial coefficients

$$\binom{p+1}{j} = \frac{(p+1)!}{j!(p+1-j)!} \tag{2.5}$$

It is obvious from (2.1) that the geometry of the curve depends not only on the control points but also the degree and the knot vector. The curve geometry can be altered by changing any such parameter. Generally in CAD models, the designer will determine the degree at the start and formulate a control polygon to represent the conceptual geometry. A suitable knot vector is implicitly created based on specific geometric features such as sharp edges, creases etc. Subsequent to the initial concept stage, the geometry is modified as required by moving the control points without necessarily changing the degree or the knot vector.

2.2. NURBS

Non Uniform Rational B-spline (NURBS) came in to prominence after Versprille [127] and Tiller [126] showed that they can unify the parametric and implicit polynomial forms that were popular for representing smooth curves and surfaces in the 1980's. Additionally NURBS are able to provide exact descriptions of all conic section such as ellipses and circles. NURBS has been an Initial Graphics Exchange Specification (IGES) standard since 1983 [104] and the flexibility and precision offered by NURBS has since made it the de facto CAD standard.

NURBS curves are a variant of B-splines that allow a more precise definition of geometry using rational basis functions. The definition for a p-th degree NURBS curve is

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u)\boldsymbol{x}_i \qquad \boldsymbol{u} = \{u_0, u_1, \ldots, u_{n+p+1}\} \tag{2.6}$$

where the basis functions $R_{i,p}(u)$ are defined as follows

$$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^{n} N_{j,p}(u)w_i} \tag{2.7}$$

Homogeneous coordinates of the form $\boldsymbol{x^w} = (wx, wy, wz, w)$ can also be used to represent a non-rational NURBS curve in a four-dimensional space

$$C^w(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{x}_i^w \qquad \boldsymbol{u} = \{u_0, u_1, \ldots, u_{n+p+1}\} \tag{2.8}$$

The weight w_i of each control point $\boldsymbol{x}_i$ is a free variable, which provides additional control over the shape of the curve.

NURBS surfaces can be constructed as the tensor product of two NURBS curves as seen in Figure 2.3. A rectangular grid of control points and two knot vectors $\boldsymbol{u}$, $\boldsymbol{v}$ are thus required.

$$\boldsymbol{u} = \{u_0, u_1, \ldots, u_{n+p+1}\}$$
$$\boldsymbol{v} = \{v_0, v_1, \ldots, v_{m+q+1}\}$$

The definition of a NURBS surface (based on the tensor product of (2.6)) of degree p in

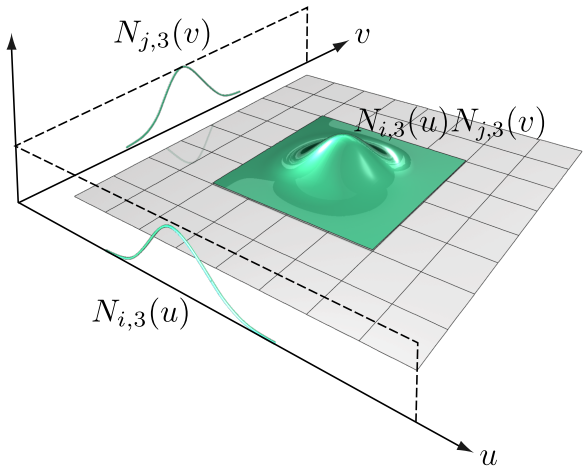


Figure 2.3.: Tensor product construction of a cubic NURBS basis function.

the u direction and q in the v direction using knot vectors $\boldsymbol{u}$, $\boldsymbol{v}$ reads

$$\zeta(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)\boldsymbol{x}_{i,j} \tag{2.9}$$

where $\boldsymbol{x}_{i,j}$ are the control points and $R_{i,j}(u)$ are the rational B-spline basis functions;

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m} N_{k,p}(u)N_{l,q}(v)w_{i,j}} \tag{2.10}$$

Similarly the tensor product of (2.8) yields a NURBS surface using homogeneous coordinates.

Knot insertion

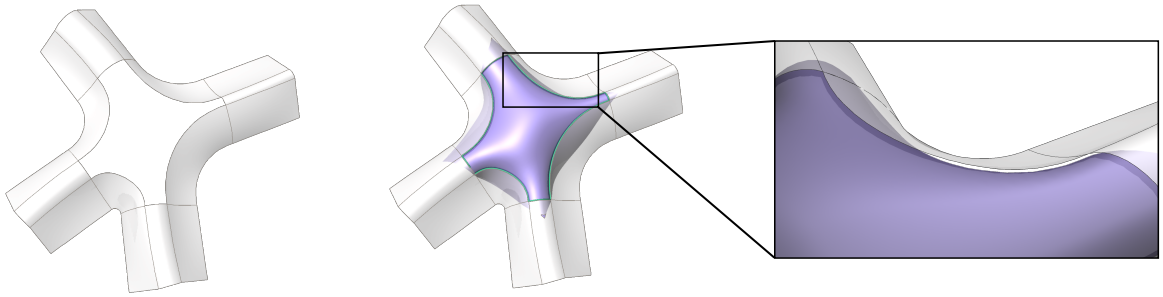
Changing mesh resolution of a NURBS geometry, i.e. adding extra control points, is achieved by a process known as knot insertion. There are many available knot insertion algorithms starting from the Casteljau algorithm [40] for evaluating Bézier curves. This algorithm was adapted by de Boor [37] for evaluating a B-spline curve at some parameter value t. These methods are based on repeatedly inserting t to the knot vector.

$$C(t) = \sum_{i=0}^{p} N_{i,p}(t)\boldsymbol{x}_i = \sum_{i=0}^{p-1} N_{i,p-1}(t)\boldsymbol{x}_i^1 = \cdots = \sum_{i=0}^{0} N_{i,0}(t)\boldsymbol{x}_i^n = \boldsymbol{x}_0^n \tag{2.11}$$

where $\boldsymbol{x}^j$ is given by an affine combination of $\boldsymbol{x}^{j-1}$. The curve is unchanged during this process but existing control points are moved and a new control point is added per each new knot inserted. Algorithms that enable multiple knots to be inserted at the same time such as the Oslo algorithm [35] are more suited for creating higher resolutions from a starting coarse control mesh as required in the present work.

Arbitrary geometry

As described in the previous section, creating multiple resolutions of a single NURBS patch is not an issue, the challenge is to perform multiresolution editing on a trimmed NURBS surface. As implied in (2.9), a NURBS surface is restricted to rectangular patches. For representing arbitrary geometry, mesh trimming is required which is simply a boolean operation that hides unwanted sections of a surface patch. In addition, the various surface patches must be stitched together to create a single smooth geometry resulting in lack of continuity. Careful positioning of the control points on either side of the trimming curve can achieve continuity only in situations where the surfaces have compatible knot vectors [17, 48]. Figure 2.4 shows an example geometry that cannot be represented using tensor-product patches. Trimming has resulted in a visually smooth surface, which on close inspection reveals the gap in the geometry. The root of the problem is that the intersection of two arbitrary NURBS patches cannot be exactly reproduced by the trimming curve [115, 9]. A remedy based on T-Splines to obtain watertight stitching of trimmed NURBS patches (with no gaps or holes) is presented in [115]. However a fine resolution mesh with T-junctions is not suitable for conventional



(a) Sixteen untrimmed NURBS patches smoothly connected.

(b) Trimmed NURBS surface and trimming curve (green). Close inspection reveals holes (inset).

Figure 2.4.: Trimming to represent non-tensor product meshes.

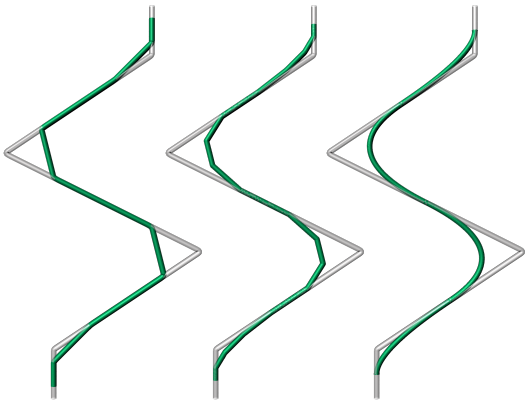


Figure 2.5.: Chaikin's [25] corner cutting example. Original curve (grey) successively refined (green) by cutting corners.

FEM solvers. Hence using a multiresolution framework based on T-splines would restrict the analysis module to isogeometric methods which also use T-splines [9].

2.3. Subdivision surfaces

Subdivision can be defined as obtaining a smooth curve or surface as the limit of a sequence of successive refinements [131]. The origins of subdivision can be dated to the corner cutting algorithm by Chaikin [25] who produced a uniform quadratic B-spline in the limit after successive refinement steps (figure 2.5). Informally, subdivision is essentially a way of obtaining B-spline curves and surfaces with uniform knots.

2.3.1. Univariate subdivision

The refinability property of uniform B-spline basis functions (2.4) can be expressed using a subdivision matrix

$$N_{i,p}(u) = \sum_{j=0}^{p+1} N_{i,p}(2u) S_{ij} \tag{2.12}$$

where the non-zero entries of the subdivision matrix S_{ij} contain the weights of the binomial coefficients defined in (2.5). Let $C(u)$ be a uniform B-spline curve of degree p initially defined with the basis $\mathbf{N}_p$ and control points $\mathbf{x}^0$. The subdivision matrix can be

used to change the basis of a curve from $\mathbf{N}_p(u)$ to $\mathbf{N}_p(2u)$ and control points from $\mathbf{x}^0$ to $\mathbf{S}\mathbf{x}^0$

$$C(u) = \mathbf{N}_p(2u)\mathbf{S}\mathbf{x}^0 \tag{2.13}$$

This process can be repeated k times to obtain an increasingly smoother control polygon as seen from Figure 2.6.

$$\begin{aligned} C(u) &= \mathbf{N}(u)\mathbf{x}^0 \\ &= \mathbf{N}(2u)\mathbf{x}^1 = \mathbf{N}(2u)\mathbf{S}\mathbf{x}^0 \\ &\quad\vdots \qquad\qquad \vdots \\ &= \mathbf{N}(2^k u)\mathbf{x}^k = \mathbf{N}(2^k u)\mathbf{S}^k\mathbf{x}^0 \end{aligned}$$

Note that for the example shown in Figure 2.6, there are twice more components in $\mathbf{N}(2u)$ than $\mathbf{N}(u)$ etc. Also, the subdivision matrices $\mathbf{S}$ gets larger during refinement. It is clear that the subdivision matrix $\mathbf{S}$ defines the following relationship between control points of two consecutive levels of refinement.

$$\mathbf{x}^{\ell+1} = \mathbf{S}\mathbf{x}^{\ell} \tag{2.14}$$

By definition, the initial control polygon is assigned level $\ell = 0$. Subdivision is easily understood when explained in the widely used cubic setting, i.e. using cubic B-splines.

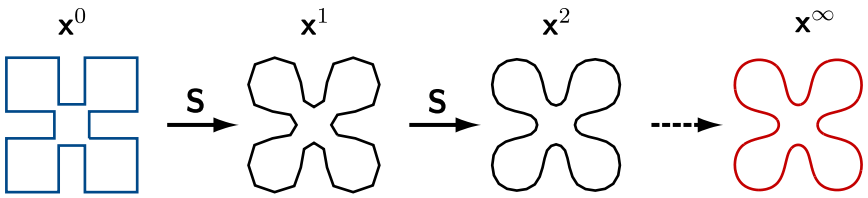
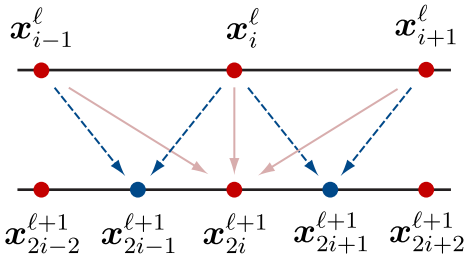
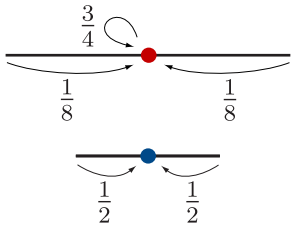


Figure 2.6.: Subdivision refinement of a cubic control polygon (shown left). The three polygons to the right are generated by repeated subdivision. Notice the increasing smoothness of the subdivided polygons.



(a) Refinement by bisecting.



(b) Masks for even (top) and odd (bottom) vertices.

Figure 2.7.: Univariate cubic subdivision refinement.

The subdivision matrix for a cubic curve has the following structure

$$\mathbf{S} = \begin{bmatrix} \dots & 0 & 0 & 0 & 0 \\ \dots & \frac{1}{8} & 0 & 0 & 0 \\ \dots & \frac{1}{2} & 0 & 0 & 0 \\ \dots & \frac{3}{4} & \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{8} & \frac{3}{4} & \dots \\ 0 & 0 & 0 & \frac{1}{2} & \dots \\ 0 & 0 & 0 & \frac{1}{8} & \dots \\ 0 & 0 & 0 & 0 & \dots \end{bmatrix}$$

Each column of the subdivision matrix is a copy of the previous column shifted down by two rows. The entries of the subdivision matrix are called *subdivision weights*. Each row contains weights defining how existing control points combine to form control points in the refined curve. These rules can be graphically shown in the form of *subdivision masks* and are widely used in computer graphics literature. It is clear that the above cubic subdivision matrix has two different type of rows and, hence, has two different masks shown in Figure 2.7b.

It is instructive to think that each subdivision step consists of a *refinement* and *averaging* steps. In the refinement step every segment of the polygon is subdivided in two segments, Figure 2.7a. Subsequently, the vertex coordinates of the refined polygon are determined

by averaging the coarse vertex coordinates with the two masks shown in Figure 2.7b.

$$\boldsymbol{x}_{2i}^{\ell+1} = \frac{1}{8}\boldsymbol{x}_{i-1}^{\ell} + \frac{3}{4}\boldsymbol{x}_{i}^{\ell} + \frac{1}{8}\boldsymbol{x}_{i+1}^{\ell} \tag{2.15a}$$

$$\boldsymbol{x}_{2i+1}^{\ell+1} = \frac{1}{2}\boldsymbol{x}_{i}^{\ell} + \frac{1}{2}x_{i+1}^{\ell} \tag{2.15b}$$

The *even vertex mask* applies to vertices that are already present in the coarse polygon and the *odd vertex mask* applies to vertices that are only present in the refined polygon. The naming odd and even for the stencils is motivated by the consecutive numbering of vertices in a polygon where newly inserted vertices receive odd numbers (2.15).

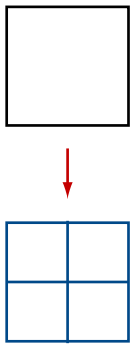
2.3.2. Bivariate subdivision

The main difference between NURBS and subdivision stems from how the univariate curves described in the previous section are adapted to create bivariate surfaces. It is now appropriate to introduce some terminology used in connection with subdivision surfaces. The *valence* denotes the number of edges connected to a vertex, vertices with valence = 4 in a quadrilateral mesh or valence = 6 in a triangular mesh are called a *regular vertex* and any non-regular vertex is an *extraordinary vertex*. Additionally the notion of control polygon is now replaced with *control mesh* to reflect the higher dimension.

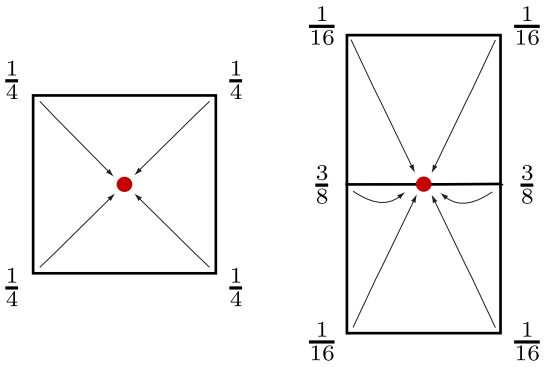
For a regular control mesh, i.e. all vertices are regular, the subdivision masks for surfaces can be constructed in the same way as NURBS using the tensor product of (2.12). In the refinement step, each quadrilateral element in the control mesh is subdivided into four quadrilaterals by introducing new vertices at the edge midpoints and element midpoints as shown in Figure 2.8a. In a cubic setting, the tensor product of (2.15a) and (2.15b) lead to the subdivision masks shown in Figures 2.8b and 2.8c.

Arbitrary geometry

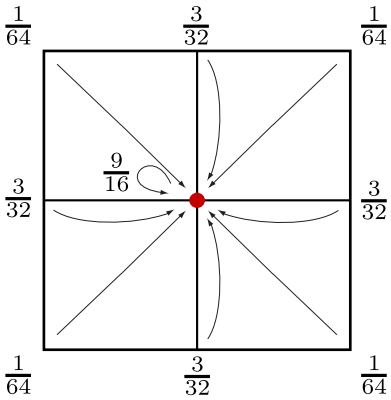
The capabilities of subdivision with respect to arbitrary geometry represented is highlighted in its popularity in character animation. A successful integration of a variant of Catmull-Clark subdivision to Pixar's Renderman [41] was used for creating the Academy



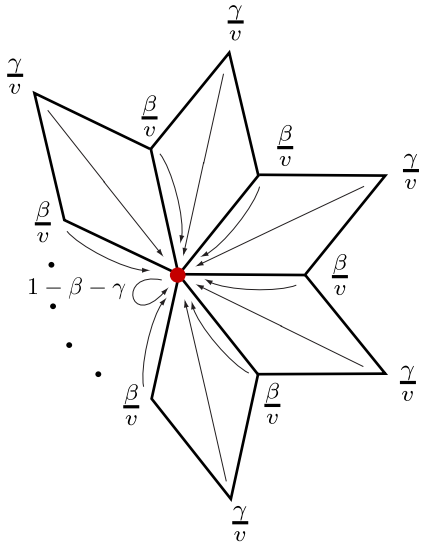
(a) Refinement by quadrisecting.



(b) Masks for odd vertices.

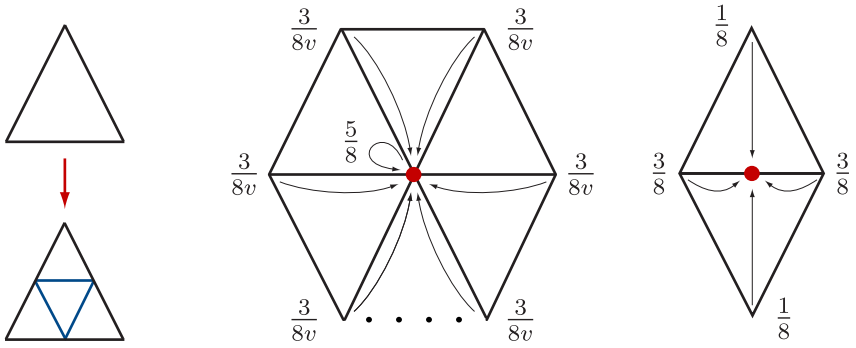


(c) Mask for regular even vertices.



(d) Valence dependent mask for non-regular even vertices.

Figure 2.8.: Bivariate Catmull-Clark subdivision refinement.



(a) Refinement by quadrisecting.

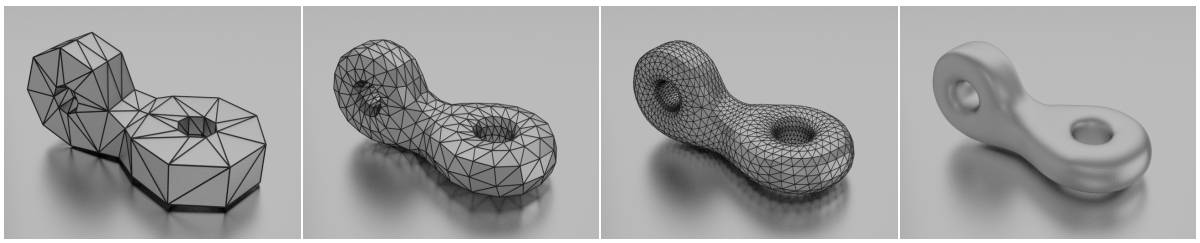
(b) Masks for even and odd vertices.

Figure 2.9.: Bivariate Loop subdivision refinement.

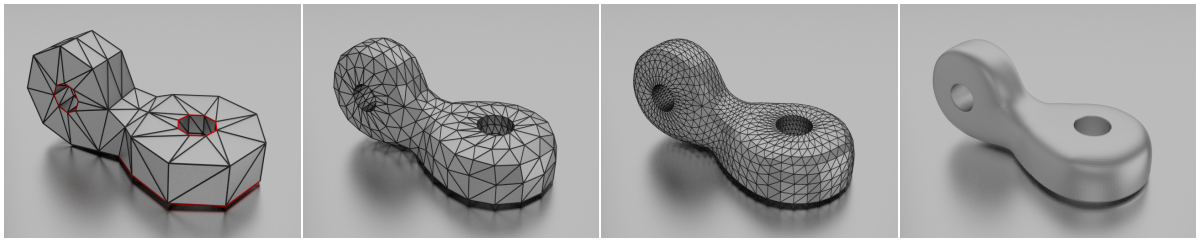
award winning short film *Geri's game*. This led to widespread use of subdivision surfaces resulting in it being the method of choice for the animation industry today.

Subdivision leads to NURBS surfaces when applied for regular meshes. The key advantage of subdivision is the unique ability of generating a single watertight smooth surface from a mesh with extraordinary vertices. An important contribution in this regard was from Catmull and Clark [21] who modified the regular cubic subdivision masks to be applied for meshes with extraordinary vertices by making them dependent of the valence of vertices. Doo and Sabin [42] provide analytical proof based on eigenanalysis of the subdivision matrix (see appendix A.1 for details). The modified mask for existing vertices presented by Catmull and Clark is shown in Figure 2.8d, with the suggested values for β and γ being $\frac{3}{2v}$ and $\frac{1}{4}$ respectively where v is the valence of the vertex. Note that the mask reduces to the one in figure 2.8c when $v = 4$ for a regular vertex.

There are many different subdivision schemes available, the most widely used of which are Catmull-Clark [21] and Loop [91] for quadrilateral and triangular meshes, respectively. Only these two subdivision schemes are used in this dissertation. Both schemes are cubic in the univariate setting but the Loop scheme yields quartic box splines in the bivariate setting for regular meshes whereas Catmull-Clark leads to cubic splines. Figure 2.9 shows the refinement pattern and the subdivision masks for Loop subdivision. More details on subdivision surfaces can be found in [111], [131].



(a) Using regular subdivision.



(b) Using extended subdivision. The edges shown in red are tagged as crease edges.

Figure 2.10.: Comparison of subdivision refinement of a connector using regular and extended Loop subdivision rules. Figures from left to right shows the control mesh, once subdivided mesh, twice subdivide mesh and a rendering of the limit surface.

Extended subdivision

The subdivision rules in (2.15) have been chosen such that the limit curve for $\ell \to \infty$ is a uniform cubic B-spline. The relaxed nature of conditions required for convergence of subdivision schemes allow room for various improvised subdivision rules to be created in order to change the interpolation and/or smoothness properties of the limit surface. Being a cubic spline the limit surface is C^2-continuous which can be reduced to C^0 continuous by modifying the subdivision weights. A comprehensive list of such extended subdivision rules for Loop and Catmull-Clark were given by Biermann et al. [14]. The present work uses these extended rules to impose surface features such as boundaries, creases etc. The methods is based on *mesh tagging* where vertices may be tagged as *creased, corner* and edges as *creased* amongst others. Special subdivision masks with modified weights are used in the vicinity of tagged vertices or edges.

Ability to represent such surface features are vital for use of subdivision in CAD where models often contain non-smooth design features. Figure 2.10 shows such an example which compares regular and extended Loop subdivision.

2.3.3. Multiresolution geometry editing

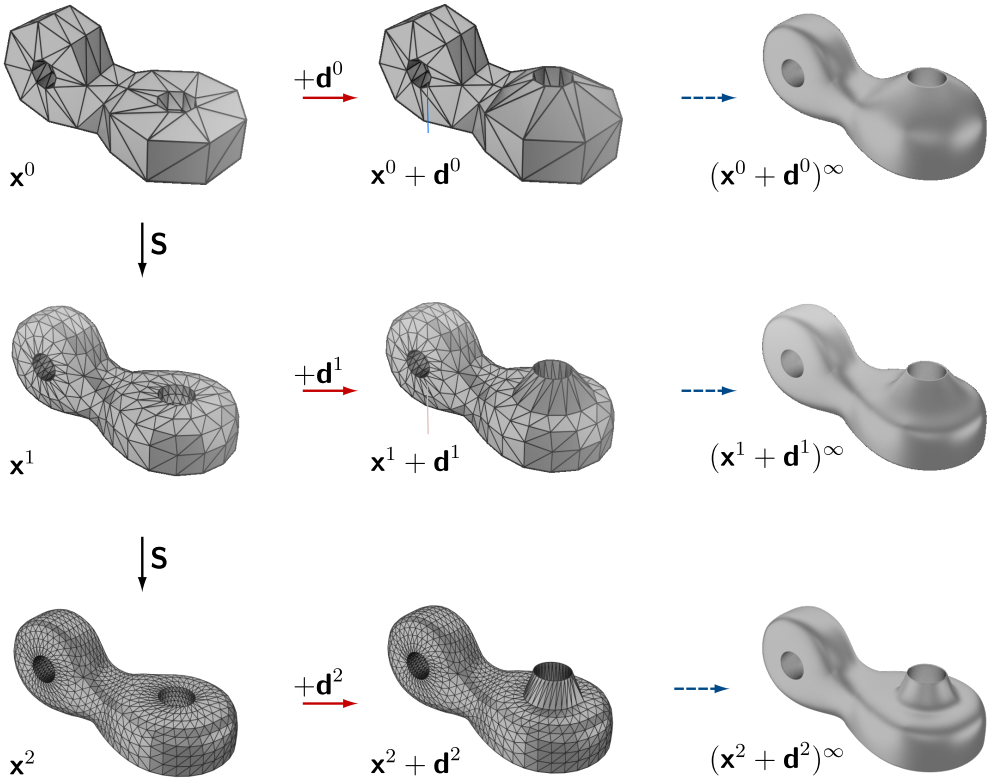


Figure 2.11.: Multiresolution editing of the connector geometry introduced in Figure 2.10a. The geometry is modified by moving the edge of one of the holes in the vertical direction. The modification is performed on levels $\ell = 0, 1, 2$ shown on the first, second and third rows respectively. Notice the effect of the modification level on the limit surface (last column).

Creation and manipulation of complex geometric models is often required in applications such as animation. As previously explained subdivision is the method of choice in animation owing to the flexibility offered in representing arbitrary geometry. Additionally, subdivision offers an elegant solution to the needs of multiresolution mesh editing semantics, namely provision of a coarse mesh that can be edited with large scale control while the resulting changes can be applied to the limit surface smoothly.

The basic idea in multiresolution editing is to modify coarse mesh coordinates to perform large-scale changes (to the limit surface) and to modify fine mesh coordinates to add localised changes. The local support property of B-splines imply that editing a control point only affects the local support of the corresponding B-spline. The local support of a control point in a coarse resolution maps to a larger region in the limit surface in

comparison to a fine resolution control point. Such geometry editing with subdivision surfaces is now standard practice in modelling packages, e.g. 3Ds Max, Maya, Blender. By way of example, this is illustrated in Figure 2.11 for the connector geometry previously introduced in Figure 2.10a using extended Loop subdivision. It can be shown that the local support or area of influence for each vertex extends over two rings of adjacent triangles in this case. First the control mesh coordinates $\mathbf{x}^0$ are modified with $\mathbf{x}^0 + \mathbf{d}^0$, where $\mathbf{d}^0$ can be thought as a user given perturbation vector. In the considered example perturbations are only applied to the vertices placed on one of the hole edges. Subsequent computation of the limit surface (by repeated subdivision) leads to a geometry with rather large scale changes. Alternatively, the edge of the hole can be perturbed on some finer level ℓ, i.e. $\mathbf{x}^\ell + \mathbf{d}^\ell = \mathbf{S}^\ell \mathbf{x}^0 + \mathbf{d}^\ell$. This results in more localised changes.

3. Progressive meshes

Constructing a multiresolution hierarchy from a given fine resolution mesh can be addressed with two alternative approaches; namely, either by creating an equivalent fine resolution mesh with subdivision connectivity or via mesh simplification. This chapter focuses on the latter where a sequence of meshes obtained from incremental mesh decimation is used to represent the geometry at different resolutions. Existing progressive mesh technologies are reviewed in view of developing a framework suitable for multiresolution optimisation.

3.1. Multiresolution modeling on arbitrary meshes

Subdivision surfaces can also be used for the bottom up approach of starting the integrated geometric design, analysis and optimisation cycle from a fine resolution model already containing many design features. However this requires a remeshing step before the model can be incorporated into the multiresolution framework. During the remeshing step, a fine resolution subdivision mesh is to be fitted to the original mesh, the parameterisation of the former needs to be such that the geometric difference with the original is minimised. Krishnamurthy and Levoy [83] provide such an example where multiple B-spline patches are fitted to arbitrary meshes. However use of subdivision surfaces in this context requires a constraint on the remeshing process; the fitted fine resolution mesh must have subdivision connectivity, i.e. semi-regular mesh without too many extraordinary vertices. Several algorithms for fitting meshes with subdivision connectivity are provided in literature; Eck et al. [43] used Voronoi diagrams and harmonic maps, the MAPS algorithms by Lee et al. [87] used vertex removal to obtain a coarse mesh on which a modified Loop scheme is used to create the parameterisation, Lee et al. [86] also use the MAPS algorithm to generate the coarse mesh but use a global optimisation strategy to adjust the control points to obtain a better correspondence with

the original. Once such a mesh with subdivision connectivity is fitted to the original, the desired multiresolution framework can be created using the inherent multiresolution structure of subdivision surfaces.

Alternatively, a different concept based on progressive meshes [69] is adopted in the present work for building a bottom-up multiresolution framework. In comparison to subdivision mesh fitting methods, progressive meshes can construct a multiresolution hierarchy from an arbitrary mesh without any remeshing.

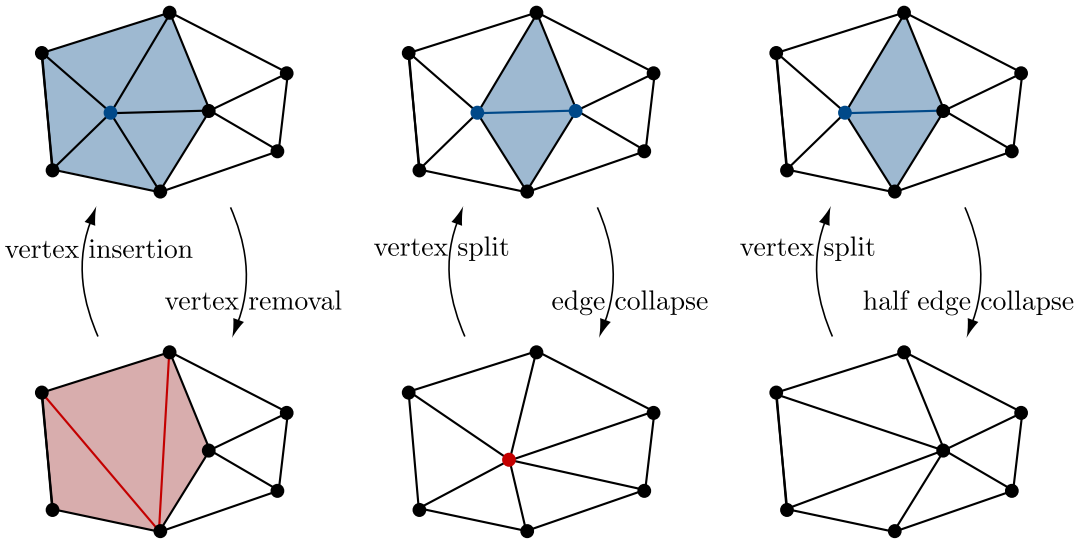
3.2. Mesh decimation

Mesh decimation has been traditionally used for reducing the complexity of high resolution meshes for data compression, see Heckbert and Garland [67], Cignoni et al. [29] for a review. Most algorithms are based on incremental mesh decimation where one vertex is removed during a single decimation step. The process can be designed to be invertible, i.e. the original mesh can be reconstructed by running the decimation scheme backwards. In this context, it is advantageous to keep a single decimation operation as simple as possible. Several available choices shown in Figure 3.1 and listed below:;

- **Vertex removal**: Delete a vertex and re-triangulate its neighbourhood.
- **Edge collapse**: Delete one edge and two adjacent triangles and merge the two involved vertices into a new position.
- **Halfedge collapse**: Takes two adjacent vertices, one is moved to the position of the other. Effectively a special case of edge collapse where the merged vertex takes the place of one of the original vertices.

In view of the need to record the decimation process for later reconstruction, it is clear that halfedge collapse is the easiest [69] from the decimation operations depicted in Figure 3.1.

The order of decimation is governed by some quality criteria which is usually a combinations of *binary* and *continuous* oracles. These quality criteria are used to keep the decimated mesh within some geometric tolerance of the original. As the name implies, a binary oracle is used to check (yes/no) if a particular decimation step violates a given



(a) Vertex removal (b) Edge collapse (c) Halfedge collapse

Figure 3.1.: Mesh decimation operations. Entities removed during decimation are shown in blue and entities added after decimation are in red.

condition, e.g. maximum aspect ratio. A priority queue is then formed from the remaining potential simplification steps according to a continuous oracle that assigns some value to the quality of the mesh after the decimation step in consideration. Note that this process requires the presence of at least one continuous oracle.

3.2.1. Quadrics

The quadric error metric by Garland and Heckbert [55] provides a way to measure the error due to a decimation step and can be used as a continuous oracle. Essentially it computes the squared distance from a vertex to the surface. The equation of a plane with unit normal $\boldsymbol{n}$ and passing through a point $\boldsymbol{p}$ is given by is $(\boldsymbol{x} - \boldsymbol{p})^T \boldsymbol{n} = 0$. Restricting the definition of a surface to a triangular mesh, i.e. a set of planes $\mathcal{P}$, the squared distance error E^s between a vertex $\boldsymbol{x}$ and the surface can be computed as the sum of

squared distances from $\boldsymbol{x}$ to each plane

$$\begin{aligned} E^s(\boldsymbol{x}) &= \sum_{i\in\mathcal{P}}((\boldsymbol{x}-\boldsymbol{p}_i)^T\boldsymbol{n}_i)^2 \\ &= \sum_{i\in\mathcal{P}}((\boldsymbol{x}-\boldsymbol{p}_i)^T\boldsymbol{n}_i\boldsymbol{n}_i^T(\boldsymbol{x}-\boldsymbol{p}_i) \\ &= \sum_{i\in\mathcal{P}}\boldsymbol{x}^T\boldsymbol{n}_i\boldsymbol{n}_i^T\boldsymbol{x} - 2(\boldsymbol{n}_i\boldsymbol{n}_i^T\boldsymbol{p})^T\boldsymbol{x} + \boldsymbol{p}_i^T\boldsymbol{p}_i \\ &= \sum_{i\in\mathcal{P}}\boldsymbol{x}^T\mathbf{A}_i\boldsymbol{x} + 2\boldsymbol{b}_i^T\boldsymbol{x} + c_i \end{aligned} \tag{3.1}$$

where $\mathbf{A}_i = \boldsymbol{n}_i\boldsymbol{n}_i^T$, $\boldsymbol{b}_i = -\mathbf{A}_i\boldsymbol{p}_i$ and $c_i = \boldsymbol{p}_i^T\boldsymbol{p}_i$. Garland and Heckbert [55] defined the following *quadric* as a triple

$$\mathbf{Q}(\mathbf{A},\boldsymbol{b},c) = \begin{bmatrix}\mathbf{A} & \boldsymbol{b}\\ \boldsymbol{b}^T & c\end{bmatrix} \tag{3.2}$$

Note that $\mathbf{Q}$ is a 4×4 matrix given that $\mathbf{A}$ is a matrix of size 3×3 and $\boldsymbol{b}$ is a vector of size 3. Essentially a quadric is a matrix that can be used to measure the squared distance error from a point to a plane. Quadrics can be added component wise; $\mathbf{Q}_i(\mathbf{A}_i,\boldsymbol{b}_i,c_i) + \mathbf{Q}_j(\mathbf{A}_j,\boldsymbol{b}_j,c_j) = \mathbf{Q}(\mathbf{A}_i+\mathbf{A}_j,\boldsymbol{b}_i+\boldsymbol{b}_j,c_i+c_j)$. The error in (3.1) can now be computed using quadrics and an augmented position vector for the vertex $\hat{\boldsymbol{x}} = [x_1\ x_2\ x_3\ 1]$

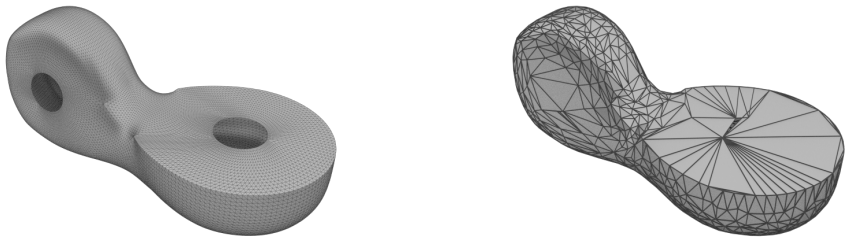
$$E^s(\boldsymbol{x}) = \sum_{i\in\mathcal{P}}\hat{\boldsymbol{x}}^T\mathbf{Q}_i^s\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}^T\left(\sum_{i\in\mathcal{P}}\mathbf{Q}_i^s\right)\hat{\boldsymbol{x}} \tag{3.3}$$

where $\mathbf{Q}^s = \mathbf{Q}^s(\boldsymbol{n}\boldsymbol{n}^T, -\boldsymbol{n}\boldsymbol{n}^T\boldsymbol{p}, \boldsymbol{p}^T\boldsymbol{p})$ is the *local approximation error quadric*, with $\boldsymbol{p}$'s given by the original positions of the vertices.

3.2.2. Feature preserving decimation of CAD geometry

Using only the local approximation error quadric $\mathbf{Q}^s$ (3.3), quantifying the squared distance error to the surface, during decimation is adequate for models with mostly curved geometry as seen, for instance, in Figure 1.7. However most CAD geometries contain flat surfaces which will give zero error for any in-plane halfedge collapses. Additionally, design features such as holes, creases etc. need to be preserved as much as possible during the decimation process.

Figure 3.2 shows an example where a CAD model with holes, creases and plane areas is



(a) Original mesh with 16638 vertices.

(b) 95% vertices decimated using $\mathbf{Q}^s$ (3.3).

Figure 3.2.: Decimation of typical CAD model containing holes, creases and plane areas using only the local approximation error quadric.

decimated using only the local approximation error quadric. The zero error in the flat region has resulted in the immediate collapse of all halfedges in this region. Clearly a priority queue based on a single error quadric is not adequate. Several types of errors can be combined with different weights as required [81] for enforcing different quality measures.

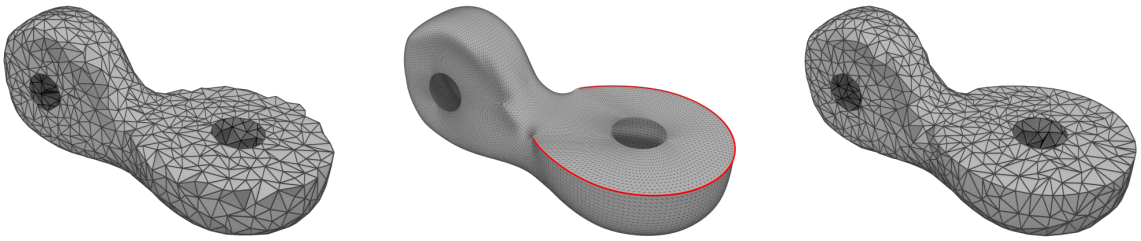
$$E(\boldsymbol{x}) = \alpha E_1(\boldsymbol{x}) + \beta E_2(\boldsymbol{x}) + \dots \tag{3.4}$$

where α, β are the weighting factors for the errors E_1, E_2. This concept can be used to combine multiple quadrics related to different error measures. For example, Garland and Zhou [56] suggests the use of a *vertex distribution quadric* for the decimation of flat regions. Let $\boldsymbol{p}$ denote the original position of a vertex with coordinate $\boldsymbol{x}$, the deviation from its original position is expressed as;

$$\begin{aligned} E^v(\boldsymbol{x}) &= (\boldsymbol{x} - \boldsymbol{p})^2 \\ &= \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{p}^T\boldsymbol{x} + \boldsymbol{p}^T\boldsymbol{p} \\ &= \hat{\boldsymbol{x}}^{\boldsymbol{T}}\mathbf{Q}^v\hat{\boldsymbol{x}} \end{aligned} \tag{3.5}$$

Where $\mathbf{Q}^v = \mathbf{Q}^v(I, -\boldsymbol{p}, \boldsymbol{p}^T\boldsymbol{p})$ is the vertex distribution quadric. Figure 3.3a shows that a more uniform decimation can be achieved by using a combination of the local approximation quadric and the vertex distribution quadric.

The next task is to preserve design details such as creases, holes etc. Garland and Heckbert [55] propose the use of a normal plane to facilitate halfedge collapse along the boundary and to constrain collapses where a vertex on the boundary is removed. For a vertex $\boldsymbol{x}$ connected to an element located along the boundary, let $\boldsymbol{e}$ denote a unit



(a) 95% vertices decimated using a combination of $\mathbf{Q}^s$ (3.3) and $\mathbf{Q}^v$ (3.5).

(b) Creased edges (red) tagged before decimation.

(c) 95% vertices decimated using a combination of $\mathbf{Q}^s$, $\mathbf{Q}^v$ and $\mathbf{Q}^e$. The latter is computed the same way as (3.6).

Figure 3.3.: Feature preserving mesh decimation for model in Figure 3.2a.

tangent vector to the boundary edge. The equation of the normal plane is $(\boldsymbol{x}-\boldsymbol{p})^T\bar{\boldsymbol{n}} = 0$ with the unit normal given by $\bar{\boldsymbol{n}} = \boldsymbol{n} \times \boldsymbol{e}$, where $\boldsymbol{n}$ is the element normal. The squared distance form this normal plane, expressed using a *boundary preserving quadric*, can be used to maintain the boundary during decimation.

$$E^b(\boldsymbol{x}) = \sum_{i \in \mathcal{B}} \hat{\boldsymbol{x}}^{\boldsymbol{T}} \mathbf{Q}_i^b \hat{\boldsymbol{x}} \tag{3.6}$$

where $\mathbf{Q}^b = \mathbf{Q}^b(\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}^T, -\bar{\boldsymbol{n}}\bar{\boldsymbol{n}}^T\boldsymbol{p}, \boldsymbol{p}^T\boldsymbol{p})$. The set $\mathcal{B}$ contains all elements connected to $\boldsymbol{x}$ adjacent to the boundary. $\mathbf{Q}^b$ needs to be computed multiple times for elements with multiple boundary edges as $\bar{\boldsymbol{n}}$ is different each time. In the present work, the same method is used to preserve any creased edges. An *edge preserving quadric* $\mathbf{Q}^e$ is generated the same way as before treating the creased edge as a boundary. For feature preserving decimation of the model from Figure 3.2a, creased edges are tagged (Figure 3.3b) before decimation resulting in the creased edges being preserved as shown in Figure 3.3c.

Incremental mesh decimation

The present work uses an algorithm for incremental decimation with halfedge collapse using quadrics originally presented by Garland [54]. The decimation is initiated by computing the quadrics for each vertex. Multiple quadrics are combined to formulate a single error quadric per vertex $\boldsymbol{x}_i$ following the approach in (3.4).

$$\mathbf{Q}_i = \mathbf{Q}^v + \sum_{j \in \mathcal{P}_i} a_j \mathbf{Q}_j^s + \sum_{m \in \mathcal{B}_i} l_m^2 \mathbf{Q}_m^b + \sum_{n \in \mathcal{E}_i} l_n^2 \mathbf{Q}_n^e \tag{3.7}$$

where the sets $\mathcal{P}_i$, $\mathcal{B}_i$ and $\mathcal{E}_i$ contain; all elements connected to vertex $\boldsymbol{x}_i$, elements with a boundary edge and elements with a creased edge, respectively. The scaling factors a and l are the area of the element and edge length, the latter is squared for dimensional consistency. Notice that in the introduced quadratics, when $\boldsymbol{p} = \boldsymbol{x}$ is used, for example $\mathbf{Q}_j^s = \mathbf{Q}_j^s(\boldsymbol{n}_j\boldsymbol{n}_j^T, -\boldsymbol{n}_j\boldsymbol{n}_j^T\boldsymbol{x}_i, \boldsymbol{x}_i^T\boldsymbol{x}_i)$, they yield to $\hat{\boldsymbol{x}}_i^T\mathbf{Q}_i\hat{\boldsymbol{x}}_i = 0$ as expected.

The list of allowable halfedge collapses is established by passing every halfedge in the mesh through binary oracles. In the present work two binary oracles are used, namely a limitation on element aspect ratio after collapse and strict preservation of user defined vertices. Next, the list of allowable halfedge collapses is sorted to formulate a *priority queue*. This is done by comparing the error in each halfedge collapse using quadrics (3.7). In each decimation step, the halfedge collapse at the top of the priority queue is performed and the priority queue is updated before the next collapse. See appendix B.1 for the exact description of the incremental decimation algorithm used in the present work.

3.3. Progressive meshes

Progressive meshes introduced by Hoppe [69] can be used for multiresolution editing of arbitrary fine input meshes as demonstrated by Kobbelt et al. [82] and Guskov et al. [59]. Essentially a sequence of intermediate meshes, generated using incremental mesh decimation introduced in Section 3.2, is used as a proxy for editing a fine resolution triangulated mesh. *Detail vectors* are used to store the difference between each successive resolution which is used to transfer the effects of global coarse mesh edits to the initial fine resolution.

Hoppe noted that the halfedge edge collapse transformation is reversible via vertex split if a record of the edge collapses is maintained. Given a starting mesh $\mathbf{x}^n$ containing n vertices, a sequence of meshes can be generated, via a series of k halfedge collapses.

$$\mathbf{x}^n \xrightarrow{\text{hcol}^1} \mathbf{x}^{n-1} \xrightarrow{\text{hcol}^2} \mathbf{x}^{n-2} \quad \cdots \quad \xrightarrow{\text{hcol}^k} \mathbf{x}^{n-k}$$

Here, each resolution contains one vertex less than the previous one. Let $\boldsymbol{x}_i^\ell$, $\boldsymbol{x}_j^\ell$ denote a pair of adjoining vertices in the current mesh $\mathbf{x}^\ell$, a halfedge collapse is represented by $(\boldsymbol{x}_i^\ell, \boldsymbol{x}_j^\ell) \rightarrow \boldsymbol{x}_i^{\ell-1}$, i.e. vertex $\boldsymbol{x}_j^\ell$ is removed during decimation and vertex $\boldsymbol{x}_i^\ell$ remains but

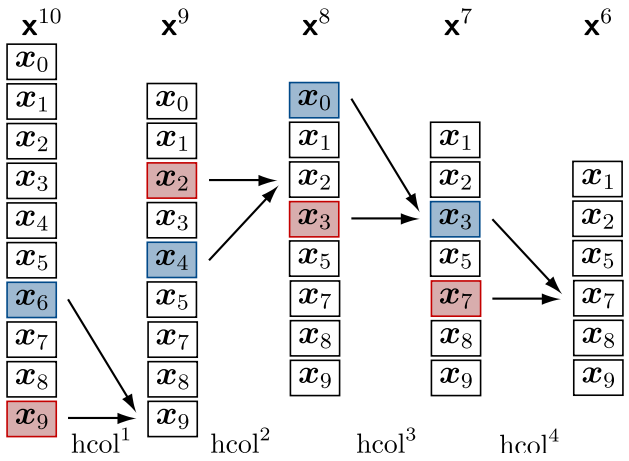
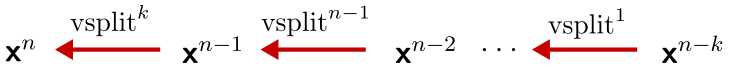


Figure 3.4.: Progressive decimation of an initial mesh with 10 vertices. Note that the superscript denoting the level of each vertex is omitted for clarity. The sequence of halfedge collapses from left to right are $(\boldsymbol{x}_9, \boldsymbol{x}_6) \rightarrow \boldsymbol{x}_9$, $(\boldsymbol{x}_2, \boldsymbol{x}_4) \rightarrow \boldsymbol{x}_2$, $(\boldsymbol{x}_3, \boldsymbol{x}_0) \rightarrow \boldsymbol{x}_3$ and $(\boldsymbol{x}_7, \boldsymbol{x}_3) \rightarrow \boldsymbol{x}_7$ respectively. In each halfedge collapse, the vertex removed is indicated in blue and the vertex retained in red.

at a coarser resolution $\boldsymbol{x}_i^{\ell-1} = \boldsymbol{x}_i^{\ell}$. A schematic diagram of progressive decimation of an initial mesh with 10 vertices is shown in Figure 3.4. The process can be run backwards and the original mesh recovered if a record of the local neighbourhood for each collapse is maintained. The limitation is that the order of restoration has to be strictly the inverse of the decimation order.



The order of half edge collapses is governed by a quadric based continuous oracle as previously explained. This defines how and when geometry features are preserved during decimation.

In addition to decimation, another important module in progressive meshes is the smoothing operator. Computing the details as the difference between two mesh resolutions, $\mathbf{d}^{\ell} = \mathbf{x}^{\ell} - \mathbf{x}^{\ell-1}$, is not suitable for transferring coarse resolution edits to finer resolutions in a smooth and intuitive manner. As indicated in (1.7), the smoothing operator affects detail restoration after the coarse resolution geometry is edited. Only certain choices for the smoothing operator lead to visually satisfactory results after detail restoration.

3.3.1. Smoothing operator

The simplest smoothing operator available is the discrete Laplacian $\mathcal{L}$ which was used by Kobbelt et al. [82] in their multiresolution mesh editing framework. However it causes severe shrinking of the mesh after repeated smoothing (Figure 3.5b). Shrinking can be avoided by using the non-shrinking two-step method by Taubin [125]. Additionally, the discrete Laplacian cannot preserve the in-plane shape of elements, i.e. element shapes change when smoothing a flat surface. Taubin [125] proposed using the inverse of the edge length as weight functions resulting in reduced in-plane shape distortions. The general format of the discrete Laplacian for smoothing a vertex $\boldsymbol{x}_i$ with the one-ring neighbourhood $\mathcal{N}_i$ [1] is as follows;

$$\mathcal{L}(\boldsymbol{x}_i) = \frac{\sum_{j \in \mathcal{N}_i} w_{ij}(\boldsymbol{x}_j - \boldsymbol{x}_i)}{\sum_{j \in \mathcal{N}_i} w_{ij}} \tag{3.8a}$$

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \mathcal{L}(\boldsymbol{x}_i) \tag{3.8b}$$

The weights w_{ij} are 1 in the general Laplacian or $1/l$ (where l is the edge length) for the modified Laplacian preserving element planar shape [125]. Note that the Laplacian (3.8) is essentially an avaraging operation very similar to that of subdivision 2.15. The non-shrinking version uses two smoothing steps

$$\check{\boldsymbol{x}}_{\boldsymbol{i}} \leftarrow \boldsymbol{x}_i + \lambda \mathcal{L}(\boldsymbol{x}_i) \tag{3.9a}$$

$$\boldsymbol{x}_i \leftarrow \check{\boldsymbol{x}}_{\boldsymbol{i}} + \mu \mathcal{L}(\check{\boldsymbol{x}}_{\boldsymbol{i}}) \tag{3.9b}$$

Initially, all vertices are smoothed using a Laplacian scaled by a parameter λ, followed by a second smoothing stage with a different scale factor μ. See Figure 3.5c for an example of non-shrinking Laplacian smoothing. A similar effect to the non-shrinking Laplacian can be achieved by simply scaling the contribution from the one-ring neighbourhood in (3.8b) using some scalar ρ_l

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \rho_l \mathcal{L}(\boldsymbol{x}_i) \quad 0 < \rho_{\mathcal{L}} \leq 1 \tag{3.10}$$

Figure 3.6 shows a comparison of different ρ_l values for smoothing the Stanford bunny model in Figure 3.5a.

[1] One-ring refers to the vertices sharing an edge with $\boldsymbol{x}_i$.

(a) Starting model, Stanford bunny with 2503 vertices.

(b) 10 smoothing steps using regular Laplacian (3.8b).

(c) 10 smoothing steps using non-shrinking Laplacian (3.9), with $\lambda = 0.6307$ and $\mu = -0.6732$.

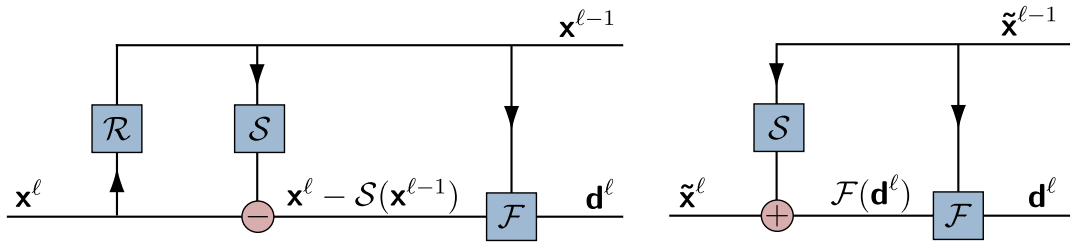
Figure 3.5.: Comparison of regular and non-shrinking Laplacian smoothing.

(a) $\rho_l = \frac{1}{3}$

(b) $\rho_l = \frac{1}{5}$

(c) $\rho_l = \frac{1}{10}$

Figure 3.6.: Use of scaled Laplacian for non-shrinking smoothing. The Stanford bunny model in Figure 3.5a is smoothed 10 steps using (3.10) with different ρ_l values.



$\mathcal{R}$: decimation $\mathcal{S}$: smoothing $\mathcal{F}$: local frame

(a) Decomposition. Initial mesh $\mathbf{x}^{\ell}$ decimated to obtain coarse resolution $\mathbf{x}^{\ell-1}$ and details $\mathbf{d}^{\ell}$.

$\mathcal{S}$: smoothing $\mathcal{F}$: local frame

(b) Reconstruction. Edited coarse resolution mesh $\tilde{\mathbf{x}}^{\ell-1}$ and original details $\mathbf{d}^{\ell}$ used to translate the editing to the original mesh.

Figure 3.7.: Wiring diagrams for computing and restoring details. Note that $\mathcal{F}$ will be described later.

3.3.2. Multiresolution editing

The multiresolution editing framework used in the present work is based on the work of Kobbelt et al. [82] and Guskov et al. [59]. The multiresolution decomposition of a fine resolution model mesh into a sequence of coarser meshes and details is termed *decomposition* and its opposite is termed *reconstruction* where details are restored. These can be conveniently expressed using *wiring diagrams* [132] as shown in Figure 3.7. Note that the diagrams in Figure 3.7 are only for a single decimation step from level ℓ to $\ell-1$. The process is repeated for multiple decimation steps.

The presence of smoothing operator $\mathcal{S}$ is essential in the reconstruction step (Figure 3.7b), where coarse level edits $\tilde{\mathbf{x}}^{\ell-1}$ are smoothed before details $\mathbf{d}^{\ell}$ are added back to restore the fine resolution. The decimation step must be designed such that its inverse gives the desired smoothing of coarse level edits. The resulting decomposition step is shown in Figure 3.7a, where details are computed as the difference between the smoothed coarse resolution and the fine resolution. Note that a *local frame* $\mathcal{F}$ is required for preserving the orientation of details and will be discussed later.

The next task is to define a suitable mesh neighbourhood for applying the smoothing operator during multiresolution editing. Kobbelt et al. [82] restricted their Laplacian smoothing operator to a user defined region around the edited coarse resolution vertex. This is not practical for multiresolution optimisation where multiple coarse resolution vertices may be edited at the same time. Guskov et al. [59] used a geometric smooth-

ing operator based on minimising the normals between two neighbouring triangles and applied this for the one-ring neighbourhood $\mathcal{N}_i^\ell$ of the coarse resolution vertex $\boldsymbol{x}_i^\ell$.

In the present work, the general Laplacian smoothing operator (3.8a) in the form (3.10) is used. Unless otherwise stated, the scaling parameter is set to $\rho_l = \frac{1}{3}$ in all presented examples. Smoothing is applied only to the one-ring of the decimated vertex, as in Guskov et al. [59], limiting the shrinking which is anyhow recovered via the details. The smoothing achieved by the non-shrinking Laplacian is not sufficient for smoothing relatively large coarse resolution edits while the geometric smoothing operator in [59] is unsuitable for geometry with plane regions.

Reconstruction step

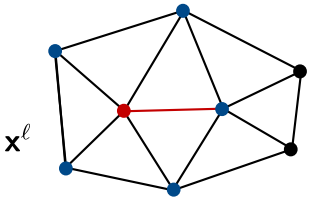
The smoothing operation in the reconstruction step ensures that the coarse resolution edits are transferred to the fine resolution in an intuitive manner. At the start of the reconstruction stage, there is a detail vector $\boldsymbol{d}_i^\ell$ available corresponding to every vertex in the fine resolution $\boldsymbol{x}_i^\ell$ to be reconstructed. The computation of the detail vector $\boldsymbol{d}_i^\ell$ will be explained later. Let $\mathcal{N}_k^\ell$ denote the set of vertices in the one-ring neighbourhood of the decimated vertex $\boldsymbol{x}_k^\ell$ (Figure 3.8a). Assume that the coordinates of the vertices in $\mathcal{N}_k^\ell$ have been edited $\boldsymbol{x}_j^{\ell-1} \rightarrow \tilde{\boldsymbol{x}}_j^{\ell-1}$ with $j \in \mathcal{N}_k^\ell$ (Figure 3.8b). The first task is to restore the decimated vertex $\boldsymbol{x}_k^\ell$ using these edited vertices (Figure 3.8c).

$$\tilde{\boldsymbol{x}}_k^\ell = \boldsymbol{d}_k^\ell + \frac{1}{v_k} \sum_{j \in \mathcal{N}_k^\ell} \tilde{\boldsymbol{x}}_j^{\ell-1} \tag{3.11}$$

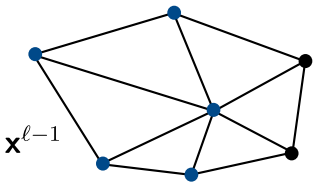
where v_k denotes the valence of vertex $\boldsymbol{x}_k^\ell$. After this restoration of the decimated vertex, the mesh is now in resolution ℓ with the vertices in $\mathcal{N}_k^\ell$ retaining their position in the coarse resolution, i.e. $\tilde{\boldsymbol{x}}_j^\ell \leftarrow \tilde{\boldsymbol{x}}_j^{\ell-1}$. Next the positions of all vertices in $\mathcal{N}_j^\ell$ are updated using Laplacian smoothing and detail restoration (Figure 3.8d).

$$\tilde{\boldsymbol{x}}_j^\ell \leftarrow \boldsymbol{d}_j^\ell + \left(\tilde{\boldsymbol{x}}_j^\ell + \rho_l \mathcal{L}(\tilde{\boldsymbol{x}}_j^\ell)\right) \tag{3.12}$$

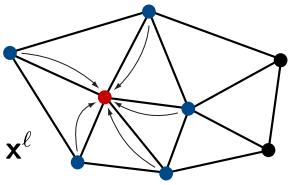
Note that the correct new positions for all vertices in $\mathcal{N}_i^\ell$ are computed before updating any to prevent dependence on the order of smoothing. Once the correct new positions are known for all vertices in $\mathcal{N}_k^\ell$, the mesh is updated.



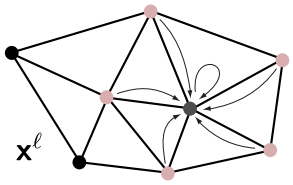
(a) In the decimation step the edge indicated in red is collapsed and the vertex in red is removed. The vertices in blue represent the one-ring $\mathcal{N}_k^{\ell}$ of the decimated vertex $\boldsymbol{x}_k^{\ell}$ (red).



(b) During mesh editing the vertices in $\mathcal{N}_k^{\ell}$ are edited in the coarse resolution; $\boldsymbol{x}_j^{\ell-1} \rightarrow \tilde{\boldsymbol{x}}_j^{\ell-1},\ j \in \mathcal{N}_k^{\ell}$.



(c) In the reconstruction step, first the decimated vertex is restored using edited vertex positions from the one-ring (3.11).



(d) Finally, in the reconstruction step, the positions of the vertices in the one-ring $\mathcal{N}_j^{\ell}$ are corrected using Laplacian smoothing (3.12).

Figure 3.8.: Mesh decimation, editing and restoration order.

Decomposition step

The procedure for computing the detail vectors can be obtained by simply reversing the previously described reconstruction process. The detail of the decimated vertex $\boldsymbol{x}_k^{\ell}$ is computed using the inverse of (3.11)

$$\boldsymbol{d}_k^{\ell} = \boldsymbol{x}_k^{\ell} - \frac{1}{v_k} \sum_{j \in \mathcal{N}_k^{\ell}} \boldsymbol{x}_j^{\ell-1} \tag{3.13}$$

Note that due to use of half edge collapse for decimation, the positions of vertices in $\mathcal{N}_k^{\ell}$ have not changed, i.e. $\boldsymbol{x}_j^{\ell-1} = \boldsymbol{x}_j^{\ell}, \quad j \in \mathcal{N}_k^{\ell}$. Next the details of the one-ring are computed by inverting (3.12)

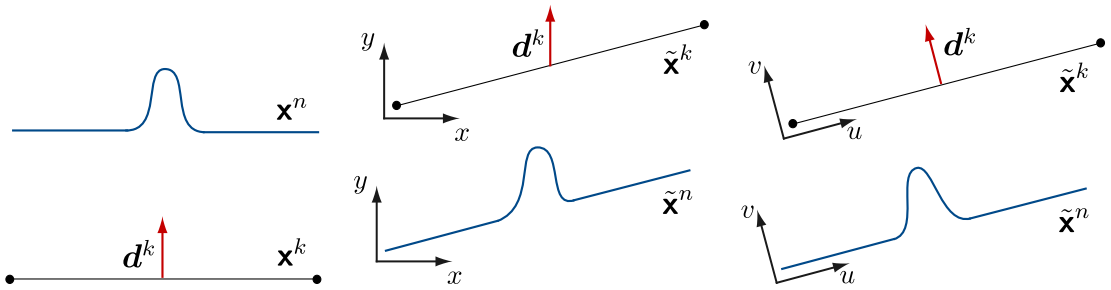
$$\boldsymbol{d}_j^{\ell} = \boldsymbol{x}_j^{\ell} - \left(\boldsymbol{x}_j^{\ell} + \rho_l \mathcal{L}(\boldsymbol{x}_j^{\ell})\right) = \rho_l \mathcal{L}(\boldsymbol{x}_j^{\ell}) \tag{3.14}$$

It is important that the positions of the vertices don't change during detail computation. Specifically, all vertices in $\mathcal{N}_k^{\ell}$ receive a smoothed position $\boldsymbol{x}_j^{\ell} \leftarrow \boldsymbol{x}_j^{\ell} + \mathcal{L}(\boldsymbol{x}_j^{\ell}), \quad j \in \mathcal{N}_k^{\ell}$ during application of (3.14). Once all details $\boldsymbol{d}_j^{\ell}$ have been computed, the vertices must

be restored to their original positions.

Local frame

The concept of local frames, introduced by Forsey and Bartels [51], is illustrated using the simple example shown in Figure 3.9. Local geometric features retain their global orientation if details are maintained in the global coordinates during decomposition and reconstruction as shown in Figure 3.9b. In most practical applications this is not desirable as it changes the global appearance of the surface. Generally it is desirable to maintain the relative orientation of such local geometric features with respect to the geometry during deformations. This can be achieved simply by storing the details with respect to a local coordinate system as demonstrated in Figure 3.9c.



(a) Original geometry $\mathbf{x}^n$ with a bump in the surface (top) is decomposed into a coarse resolution $\mathbf{x}^k$ and a single detail vector $\boldsymbol{d}^k$.

(b) Detail described in global coordinate system. Relative orientation of feature is not preserved after reconstruction (bottom).

(c) Detail maintained in local coordinate system. Relative orientation of feature is preserved after reconstruction.

Figure 3.9.: Local frames for maintaining orientation of details. The geometry is deformed $\mathbf{x}^k \rightarrow \tilde{\mathbf{x}}^k$ before reconstruction.

Local frames can be easily constructed as vertex based, see Kobbelt et al. [82] for details. Consider decomposition and reconstruction for the halfedge collapse $(\boldsymbol{x}_i^\ell, \boldsymbol{x}_j^\ell) \rightarrow \boldsymbol{x}_i^{\ell-1}$. The local frame must be defined using coarse resolution geometry $\mathbf{x}^{\ell-1}$ in view of (3.13) for restoring the detail of the decimated vertex. In the present work, $\boldsymbol{x}_i^{\ell-1}$ is selected as the origin of the local coordinate system. Alternatively a different vertex from the one-ring of $\boldsymbol{x}_j^\ell$ can be used such that the detail length is minimised.

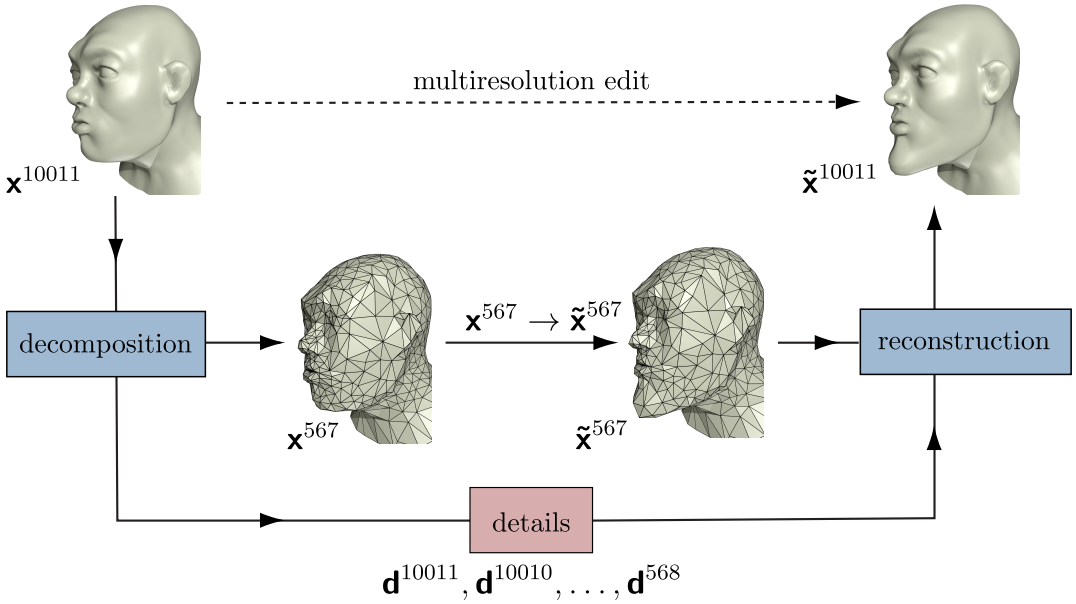
Let $\boldsymbol{w}$ denote the average unit normal of all elements connected at $\boldsymbol{x}_i^{\ell-1}$ and $\boldsymbol{e}$ be a unit vector along an edge connected to $\boldsymbol{x}_i^{\ell-1}$. A local frame can be constructed with direction

vectors $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ where $\boldsymbol{u} = \boldsymbol{e} \times \boldsymbol{w}$ and $\boldsymbol{v} = \boldsymbol{u} \times \boldsymbol{w}$. During decomposition (3.13), (3.14) the detail $\boldsymbol{d}_i^\ell$ is transformed into the local frame. It is transformed back during the reconstruction stage (3.11), (3.11).

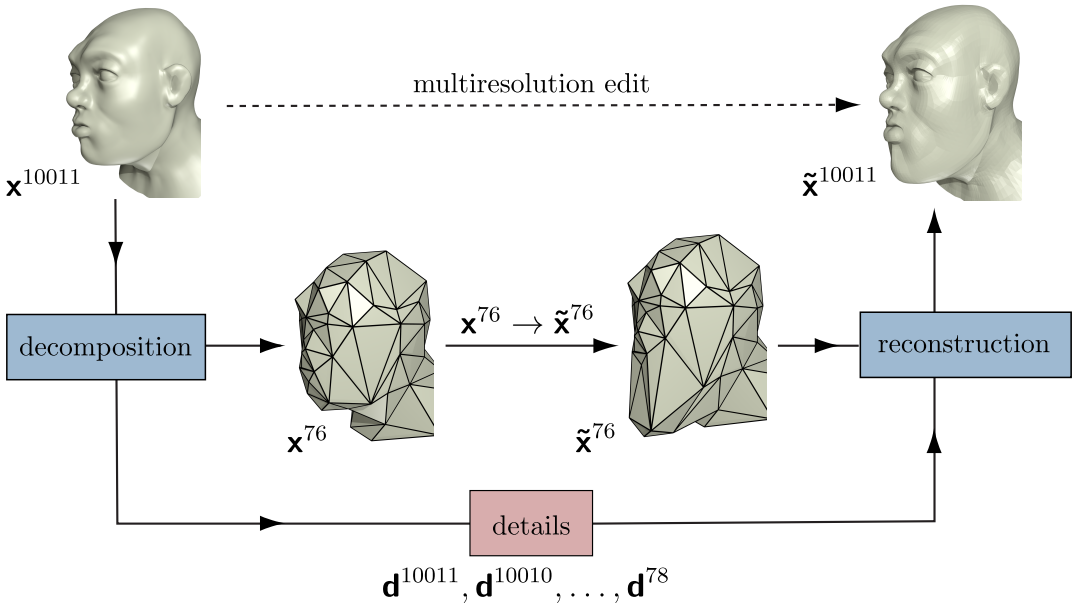
Examples

A frequently used multiresolution example in the computer graphics community is manipulating facial features of a mannequin surface mesh. A similar example in Figure 3.10 demonstrates the interplay of the decomposition (Figure 3.7a) and reconstruction (Figure 3.7b) procedures to obtain multiresolution editing. In Figure 3.10a the geometry with 10011 vertices is edited after 9444 decimation steps resulting in a more local shape change compared to Figure 3.10b where editing is done after 9935 decimation steps.

A multiresolution study of a plate with localised surface features is made in Figure 3.11. This example highlights several features required of the bottom-up multiresolution shape optimisation framework proposed in this dissertation; presence of plane geometry, globalisation of shape changes with multiresolution edits and maintaining orientation of local surface features. The geometry is initially a rectangular plate of 6×12 with 2849 vertices. Localised surface perturbations are added along the normal for all vertices within a circular region around the centroid. The initial model $\mathbf{x}^{2849}$ is decimated to coarse resolutions $\mathbf{x}^{227}$ and $\mathbf{x}^{9}$ with 2622, and 2840 halfedge collapses respectively. During decimation, the vertex at the centroid is preserved using a binary oracle and is later deformed by adding a unit displacement along the normal direction as indicated in Figure 3.11a. Restoration of this geometry edited at different resolutions results shape changes with varying degrees of support as evident from the elevation colour contours in 3.11b and 3.11c. Additionally, it is clear from the side view that in each of the restored geometries the local surface perturbations remain normal to the surface of the plate.

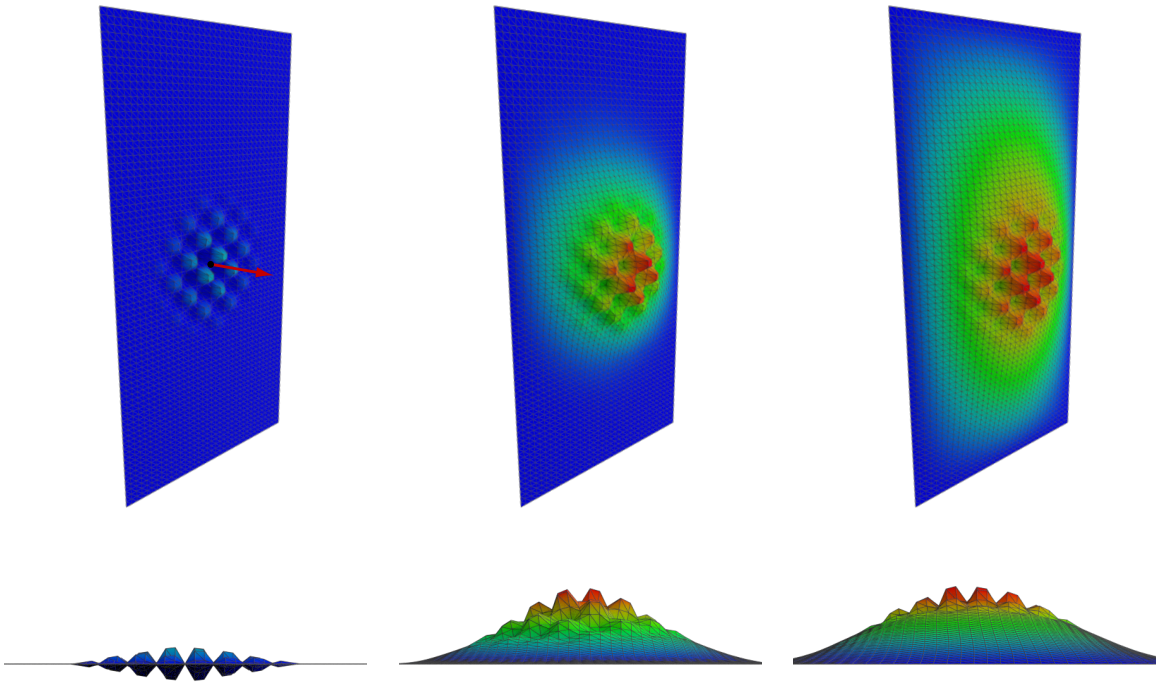


(a) Edited after 9444 halfedge collapses.



(b) Edited after 9935 halfedge collapses.

Figure 3.10.: Multiresolution editing of mannequin head. Initial mesh contains 10011 vertices. Geometry is edited after decomposition followed by reconstruction for transferring the coarse resolution edits to the original fine resolution. In the top figure the mesh level $\ell = 567$ and in the bottom figure the level $\ell = 76$ are edited.



(a) Initial geometry $\mathbf{x}^{2849}$ (b) Final geometry after editing coarse mesh $\mathbf{x}^{85}$ (c) Final geometry after editing coarse mesh $\mathbf{x}^{9}$

Figure 3.11.: Multiresolution editing study of plate with surface features. The initial mesh containing 2849 vertices is edited at different coarse resolutions. The colour contours indicate elevation from the plate surface. The top and bottom rows show respectively an isometric and a side view of the fine resolution geometry.

4. Multiresolution geometry representation

This chapter provides a detailed explanation of how the multiresolution geometric representations presented in Chapters 2 and 3 can be used for multiresolution shape optimisation.

4.1. Multiresolution shape optimisation

As discussed in Chapter 1, the motivation for using multiresolution geometry for shape optimisation is better integration of geometric modelling and finite element analysis. The parameter-free optimisation methods mostly lead to too many design degrees of freedom and to wiggly shapes unsuitable for design [18, 95, 66]. Ideally it is desirable to move the mesh while preserving the regularity of the geometric boundary and the finite element mesh. This can be achieved by using blending functions, such as B-spines [18] or Bézier curves [66], Laplacian smoothing [95] and describing the boundary as a collection of primitive geometries (lines, curves, arcs etc) [74]. Overall, the goal is to keep the number of design variables significantly smaller than the number of nodal coordinates in the finite element mesh. See Haftka et al. [63] for a survey of geometry description techniques in shape optimisation.

In topology optimisation, material distribution in a design space is optimised, the issue of oscillatory optimised solutions has been remedied by various filtering methods, see Sigmund and Petersson [119], Bendsøe and Sigmund [12] for details. Such filters essentially recover a desirable solution by removing oscillations with a wavelength below a specified filter radius. Similar filtering methods can be adopted for parameter-free shape optimisation [36, 15]. The drawback of the filtering approaches is the dependency

of the final solution on the filter used. To achieve the same objective as the filtering approaches in the present work, the resolution difference between analysis and design geometry (Figure 1.5) is used for obtaining smooth regulated shape changes. In addition to regularising the optimisation process, the proposed multiresolution framework provides the connectivity between the different modules involved, namely design, analysis and optimisation, creating an integrated product development cycle.

The use of multiple resolutions have previously been demonstrated in structural topology optimisation by employing the microstructure approach with a wavelet based variable space [80, 106, 96]. Quan [108] used a B-spine based density filter in his microstructure approach to topology optimisation. In terms of multiresolution CAD geometry, Cervera and Trevelyan [24] added more control points to NURBS curves via knot insertion during isogeometric topology optimisation, increasing the size of the design space. Kiendl [79] noted in his NURBS based isogeometric shape optimisation framework that the resolution of the mesh containing the design variables can be refined for optimising more localised design features. Although NURBS geometry resolution can be locally changed by adding and removing of control points via knot insertion or T-splines, subdivision surfaces offer a natural framework for multiresolution geometry representation. Similarly, methods based on mesh decimation can facilitate the creation of multiple resolution from a given fine resolution design model as demonstrated in Chapter 3.

To restate the methodology outlined in Section 1.2, two geometry resolutions are maintained; a fine resolution representing the optimised shape and a coarse resolution with the design variables. Analysis is done using the fine resolution geometry which is updated according to optimisation of the coarse resolution design variables. In addition to initial construction of the multiresolution hierarchy, the process requires two types of data transfer between different resolutions. Geometry edits need to be transferred from coarse to fine resolution and the design sensitivities from fine to coarse. The latter requires better understanding about the computation and discretisation of design sensitivity in a multiresolution context.

4.2. Design sensitivity

Design sensitivity is defined as the partial derivative of a cost or objective with respect to a design variable (1.2). In the following, the first task is to clarify the notion of a

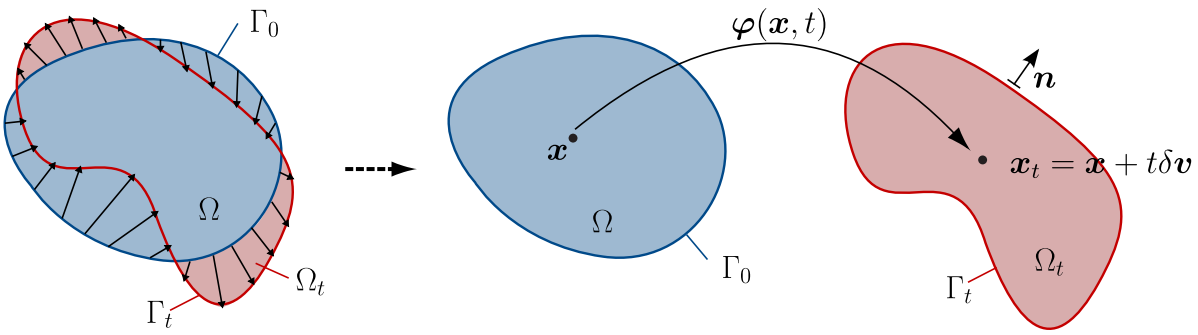


Figure 4.1.: Geometry mapping between initial design configuration Ω and current design configuration Ω_t.

design variable. The design space in shape optimisation is defined by all the allowable perturbations of the geometric shape. In this context, a linear mapping which maps a given domain Ω into a perturbed domain Ω_t needs to be defined. With this mapping, a material point with the coordinate $\boldsymbol{x} \in \Omega$ is mapped onto

$$\boldsymbol{x}_t = \boldsymbol{x} + t\delta\boldsymbol{v}, \quad t \geq 0 \tag{4.1}$$

where $\delta\boldsymbol{v}$ is a prescribed constant vector field and t is a scalar parameter. In the usual continuum mechanics terminology, Ω is the reference configuration, Ω_t is current configuration, $\delta\boldsymbol{v}$ is the prescribed velocity vector and t is the (pseudo-) time. In shape optimisation literature the mapping (4.1) is usually expressed as

$$\Omega_t = (\boldsymbol{I} + t\delta\boldsymbol{v})\Omega \tag{4.2}$$

with $\boldsymbol{I}$ the identity mapping. The optimisation problem (1.1) can be restated in the new setting (Figure 4.1)

$$\text{minimise} \qquad \mathcal{J}(\Omega, \boldsymbol{u}(\boldsymbol{x})) \tag{4.3a}$$

$$\text{such that} \quad g_i(\Omega, \boldsymbol{u}(\boldsymbol{x})) = 0 \quad i = 1, \ldots, n_g \tag{4.3b}$$

where $\boldsymbol{u}(\boldsymbol{x})$ is the state variable, i.e. displacement field. Solving (4.3) using gradient based mathematical programming methods requires the directional derivative of the objective function in the direction of the velocity field $\delta\boldsymbol{v}$

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\Omega}(\Omega, \boldsymbol{u}(\boldsymbol{x}))\delta\boldsymbol{v} = \lim_{t\to 0} \frac{\mathcal{J}(\Omega_t, \boldsymbol{u}(\boldsymbol{x}_t)) - \mathcal{J}(\Omega, \boldsymbol{u}(\boldsymbol{x}))}{t} \tag{4.4}$$

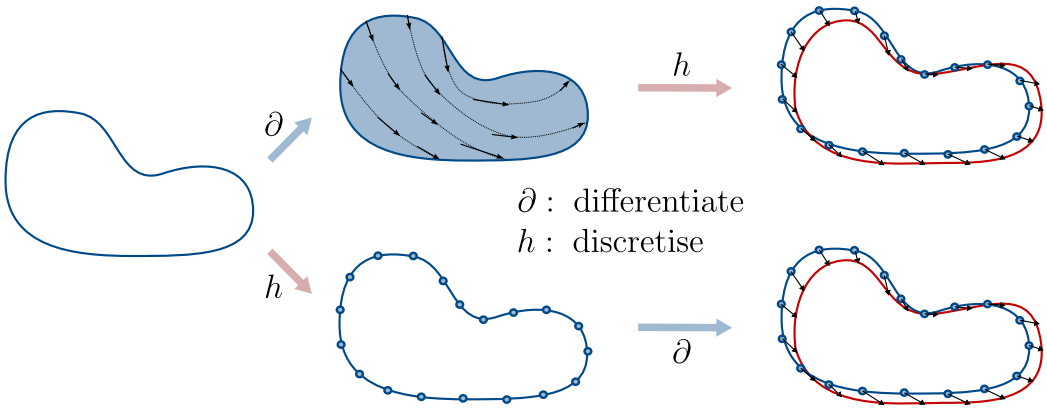


Figure 4.2.: Discrete (bottom) and continuum (top) approaches in design sensitivity analysis. The shape change is given by the velocity field $\boldsymbol{V}$ indicated as black arrows.

Hereafter this derivative will be denoted by $D\mathcal{J}(\Omega, \boldsymbol{u}(\boldsymbol{x}))[\delta \boldsymbol{v}]$,

There are two different approaches (Figure 4.2) to shape design sensitivity analysis based on the order of discretisation and differentiation of (4.4) and governing equations. In a discrete setting, differentiation is done with respect to a set of discrete design variables that define the shape. Alternatively, in a continuum setting differentiation is done with respect to the shape itself and the resulting shape gradient later discretised among design variables. The two methods are theoretically equivalent under certain conditions which are generally violated when computed numerically [27, 129]. Refer to Haftka et al.[2, 62] for a review of different methods for sensitivity analysis.

In the present work, Chapter 5 uses the discrete approach for shape optimisation of shell structures, whereas Chapter 6 uses the continuum approach for shape optimisation of solids. In both cases, the optimisation problem is formulated for the fine resolution mesh. Hence the shape derivative is obtained as a vector quantity at each node in the fine resolution mesh. In this respect, fine resolution nodes can be treated as pseudo design variables. The multiresolution geometry framework is expected to transfer the sensitivity vectors from these pseudo design nodes in the fine resolution to the coarse resolution design nodes. The optimisation problem is then solved in the coarse resolution and the geometry is updated accordingly.

Note that it is also possible to compute the shape derivative with respect to the coarse resolution design variables where the multiresolution framework is embedded in the

solver module. However this restricts the use of different geometric frameworks and solver modules and will not be used in the present work.

4.3. Top-down multiresolution framework

One essential component of the proposed multiresolution subdivision framework (Figure 1.5) is the coarsening method required to project design sensitivity data from fine to coarse resolutions. As discussed in Section 4.2, design sensitivities are computed as a vector quantity in the fine resolution mesh and need to be converted to vector quantities in the coarse resolution mesh. The framework for transfer of data between multiple resolution required in the present work is inherently linked to wavelets. A brief outline of wavelet methods are next presented for better understanding of data transfer between multiple resolutions.

4.3.1. Wavelets

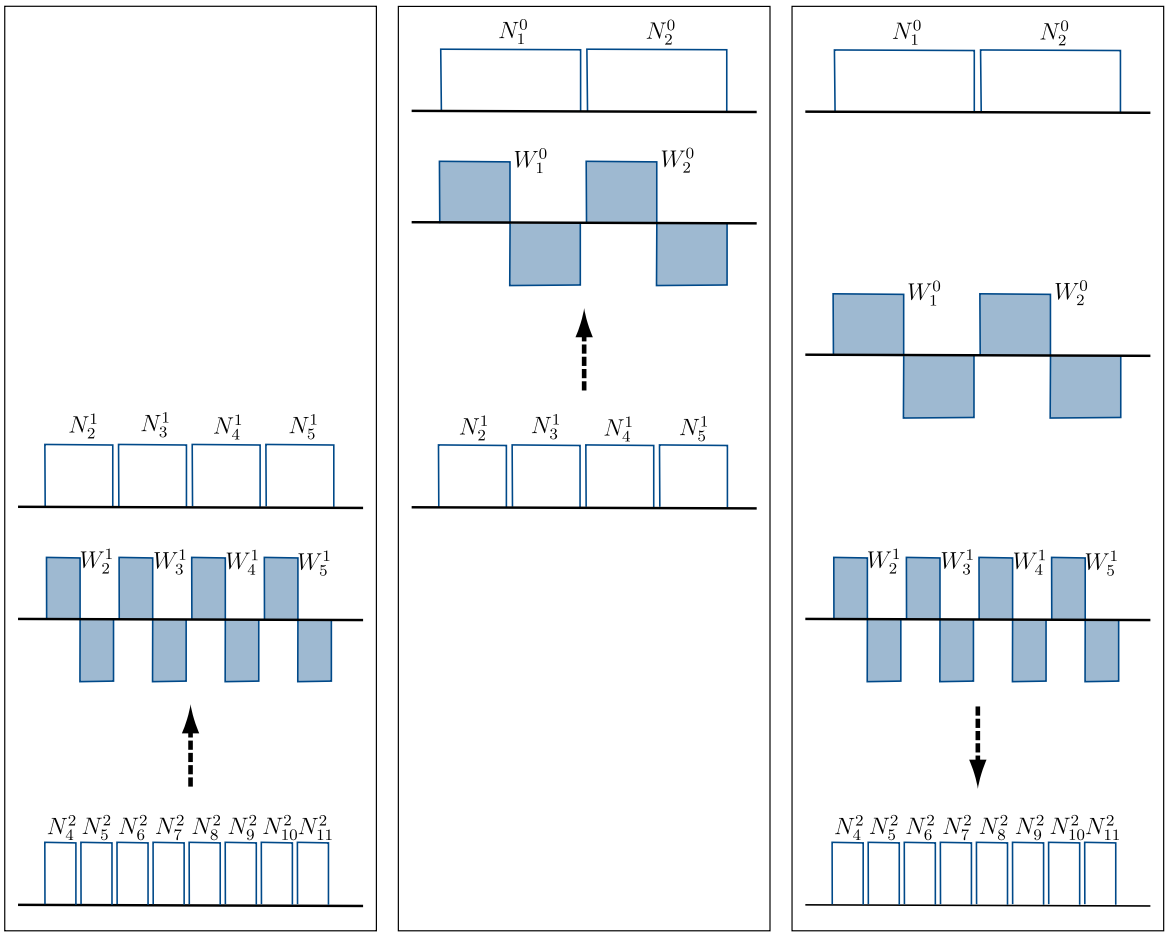
Wavelets offer a natural choice for hierarchical representation of functions. Finkelstein et al. [49] used B-spine wavelets for multiresolution editing of curves and the relationship between wavelets and subdivision was presented by Lounsbery et al. [92]. See Stollnitz et al. [122] and Schröder [112] for more details on wavelets. The wavelet concept is demonstrated here using Figure 4.3 which shows the Haar wavelet [61], the most basic wavelet basis. Let $\mathbf{x}^\ell$ be the control points of a degree 0 uniform B-spine (box functions) curve at level ℓ (c.f. Equation 2.13)

$$C(u) = \sum_i N_i^\ell(2u^\ell)\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{\ell}}, \quad \boldsymbol{x}_i^\ell \in \mathbf{x}^\ell \tag{4.5}$$

The variable u denoting the parametric coordinate will be omitted for brevity hereafter, i.e. $N^l = N_i^l(2l)$. The resolution of a basis function is indicated by the superscript. The Haar wavelet can be used to construct a hierarchical decomposition of the curve as follows;

$$\boldsymbol{x}_i^{\ell-1} = \frac{\boldsymbol{x}_{2i}^\ell + \boldsymbol{x}_{2i+1}^\ell}{2}, \qquad \boldsymbol{x}_i^{\ell-1} \in \mathbf{x}^{\ell-1} \tag{4.6a}$$

$$\boldsymbol{d}_i^{\ell-1} = \boldsymbol{x}_{2i}^\ell - \boldsymbol{x}_{2i+1}^\ell, \qquad \boldsymbol{d}_i^{\ell-1} \in \mathbf{d}^{\ell-1} \tag{4.6b}$$



(a) Basis functions $\mathcal{N}^2$ replaced by $\mathcal{N}^1$ and $\mathcal{W}^1$.

(b) Basis functions $\mathcal{N}^1$ replaced by $\mathcal{N}^0$ and $\mathcal{W}^0$.

(c) Reconstruction of the original basis.

Figure 4.3.: Haar wavelet example from Gortler [58]. Basis of degree 0 B-spine curve $C = \sum N^2 \boldsymbol{x}^2$ is transformed to a Haar wavelet basis.

The process can a reapplied to $\mathbf{x}^{\ell-1}$ yielding $\mathbf{x}^{\ell-2}$ and $\mathbf{d}^{\ell-2}$. This can be repeated up to level 0. The original curve can now be reconstructed without loss of information using *wavelet decomposition*

$$C = \sum_i N_i^0 \boldsymbol{x}_i^0 + \sum_{m=0}^{\ell-1} \sum_i W_i^m \boldsymbol{d}_i^m, \quad N_i^0 \in \mathbf{N}^0,\ W_i^m \in \mathbf{W}^m \tag{4.7}$$

where $\mathbf{N}^0$ and $\mathbf{W}^m$ are the B-spine and wavelet shape functions at level 0 and m respectively. The key idea is to replace a fine level function space with a coarse level space and the difference.

$$\mathbf{N}^\ell = \mathbf{N}^{\ell-1} + \mathbf{W}^{\ell-1} \tag{4.8}$$

4.3.2. Subdivision coarsening

The Haar wavelet represents subdivision coarsening by (4.6a) which is essentially a data fitting problem. The fine resolution control points $\mathbf{x}^{\ell+1}$ at level $\ell+1$ are known and we seek a mapping $\mathbf{R}$ to find the coarse resolution control points $\mathbf{x}^{\ell}$ at level ℓ (1.4). The difference between the two resolutions are stored in details as done by the Haar wavelet in (4.6b). Similarly, details can be simply computed as the difference between the fine resolution control points and once subdivided coarse resolution control points [113].

$$\begin{aligned}\mathbf{d}^{\ell+1} &= \mathbf{x}^{\ell+1} - \mathbf{S}\mathbf{x}^{\ell} \\ &= \mathbf{x}^{\ell+1} - \mathbf{S}\mathbf{R}\mathbf{x}^{\ell+1} \\ &= (\mathbf{I} - \mathbf{S}\mathbf{R})(\mathbf{x}^{\ell+1})\end{aligned} \tag{4.9}$$

Note that the details need to be converted via local frames (Figure 3.9) to maintain orientation during multiresolution editing. Ideally the magnitude of the detail vectors are kept as short as possible which is necessary to prevent undesirable geometric effects when details are restored after editing the coarse resolution. The magnitude of the details is an indirect measure of the difference between the two resolutions.

In the present work, in addition to projecting from fine to coarse, the subdivision coarsening $\mathbf{R}$ is required to remove high frequency noise from the shape derivative computed in the fine resolution, i.e. low pass filtering. The filtering frequency is implicitly related to the distance between control points in coarse and fine resolutions. Ideally the filter should only remove noise with frequency matching the fine resolution. For example, consider some vector field is passed down to the fine resolution using (2.14). If this is projected back to the coarse resolution without applying any modifications in the fine resolution, the original vector field is expected to be recovered. In order to have such behaviour the coarsening method needs to be the inverse of subdivision, i.e. the details need to be zero when the fine resolution is the once subdivided coarse resolution $\mathbf{x}^{\ell+1} = \mathbf{S}\mathbf{x}^{\ell}$.

$$\begin{aligned}\mathbf{0} &= \mathbf{S}\mathbf{x}^{\ell} - \mathbf{S}\mathbf{R}\mathbf{S}\mathbf{x}^{\ell} \\ \mathbf{S}\mathbf{R}\mathbf{S}\mathbf{x}^{\ell} &= \mathbf{S}\mathbf{x}^{\ell} \\ \mathbf{R}\mathbf{S} &= \mathbf{I}\end{aligned} \tag{4.10}$$

This property will be referred to as *identity property* hereafter.

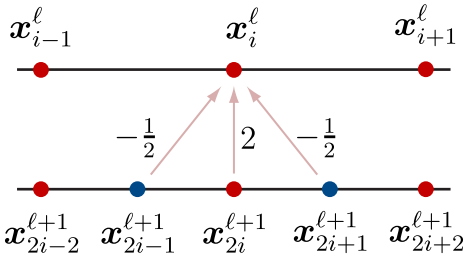


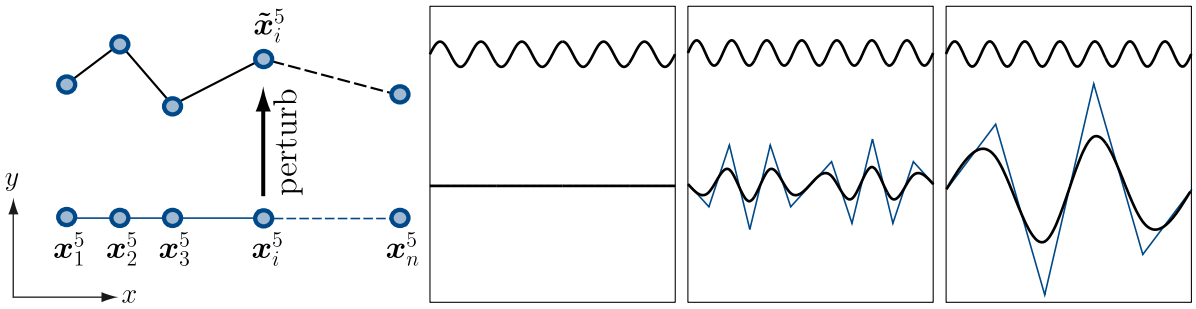
Figure 4.4.: Cubic reverse subdivision relations.

4.3.3. Coarsening methods

There are various methods available for determining a subdivision coarsening operator $\mathcal{R}$, the simplest of which is sub-sampling. Although capable of removing some noise, the identity is not maintained. Having a local coarsening operator is beneficial due to convenience of application. The inverse of the subdivision mask [85] or the limit mask [88] can provide two families of such methods. The drawback of these is that the exact average of the fine resolution geometry is not always maintained. Bi-orthogonal wavelets [34] for one-dimensional problems avoid this problem but are complicated to both implement and to generalise for arbitrary connectivity meshes. A much simpler method is to use least squares fitting [70, 13], the disadvantage being having to solve a global linear system. There are other methods that use a combination of the above to obtain better coarsening operators. Constrained interpolation by Halstead et al. [65] solves a linear system to get Catmull-Clark control points interpolating a given mesh. The control points are next subdivided and optimised according to a fairness norm.

Reverse subdivision

It is known that since subdivision is a smoothing operation its reverse may create a high-energy coarse approximation [85]. The coarsening operator $\mathbf{R}$ can be obtained by locally inverting the refinement equation (2.14) applied at a vertex. Essentially the vertex masks (the mask for refining even vertices, examples of which are shown in Figures 2.8c and 2.8d for Catmull-Clark subdivision) are inverted. Reverse subdivision satisfies the identity requirement (4.10) and yields local reverse subdivision masks that can be conveniently applied. To derive the reverse subdivision relations, consider a single



(a) Perturbation (b) $n = 13$, $f = 12\pi$ (c) $n = 13$, $f = 14\pi$ (d) $n = 6$, $f = 14\pi$

Figure 4.5.: Inconsistency of reverse subdivision. A cubic curve with n vertices and length 1 is subdivided 5 times and perturbed by adding a sinusoidal noise. This means for a vertex with index i the perturbation is of the form $\tilde{\boldsymbol{x}}_i{}^5 = \boldsymbol{x}_i{}^5 + \boldsymbol{e}_1 a \sin(f \boldsymbol{x}_i^5 \cdot \boldsymbol{e}_0)$ where $\boldsymbol{e}_0$, $\boldsymbol{e}_1$ are the basis vectors $(1,0)^\mathsf{T}$, $(0,1)^\mathsf{T}$, a is the amplitude and f is the angular frequency. The perturbed curve is coarsened $\tilde{\boldsymbol{x}}_i{}^0 = \mathbf{R}^5 \tilde{\boldsymbol{x}}_i{}^5$ using five reverse subdivision steps. Three different cases with different n, f values and constant amplitude $a = 0.05$ are shown. In each example, the solid black lines indicate the limit curve before (top) and after (bottom) coarsening. The blue line represents the control polygon after coarsening.

subdivision step of the one-ring neighbourhood of a control point $\boldsymbol{x}_i^\ell$ (c.f. Figure 2.7)

$$\mathbf{x}^{\ell+1} = \mathbf{S}\mathbf{x}^\ell$$

$$\begin{bmatrix} \boldsymbol{x}_{2i-1}^{\ell+1} \\ \boldsymbol{x}_{2i}^{\ell+1} \\ \boldsymbol{x}_{2i+1}^{\ell+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{i-1}^{\ell} \\ \boldsymbol{x}_{i}^{\ell} \\ \boldsymbol{x}_{i+1}^{\ell} \end{bmatrix}$$

The subdivision matrix for this neighbourhood can simply be inverted to obtain the coarsening operator

$$\mathbf{x}^{\ell} = \mathbf{S}^{-1}\mathbf{x}^{\ell+1}$$

$$\begin{bmatrix} \boldsymbol{x}_{i-1}^{\ell} \\ \boldsymbol{x}_{i}^{\ell} \\ \boldsymbol{x}_{i+1}^{\ell} \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & -2 & 0 \\ -\frac{1}{2} & 2 & -\frac{1}{2} \\ 0 & -2 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{2i-1}^{\ell+1} \\ \boldsymbol{x}_{2i}^{\ell+1} \\ \boldsymbol{x}_{2i+1}^{\ell+1} \end{bmatrix}$$

A major shortcoming of reverse subdivision is the lack of consistency in coarsening when details are present, an examples of which is shown in Figure 4.5. This is a result of the coarsening operator $\mathbf{R} = \mathbf{S}^{-1}$ amplifying certain frequencies.

B-spine wavelets

The Hierarchical B-spines approach to multiresolution modelling by Forsey and Bartles [51] uses a hierarchy of nested B-spine at different resolutions for editing spline curves at multiple resolutions. The B-spines at different resolutions are related by (2.4). It is clear that maintaining multiple B-spine resolutions concurrently leads to an over-representation. In comparison, wavelets use a basis where only the difference between adjoining resolutions is present (4.8). B-spine wavelets [58], are a class of wavelets that fills the gap between adjoining B-spine resolutions. Consider uniform cubic B-spines with the following refinement relationship

$$\mathbf{N}^{\ell} = \mathbf{S}\mathbf{N}^{\ell+1} \tag{4.11}$$

where $\mathbf{S} = \left\{\frac{1}{8}, \frac{1}{2}, \frac{3}{4}, \frac{1}{2}, \frac{1}{8}\right\}$. Similarly cubic B-spine wavelets $\mathbf{W}$ have the following refinement relationship [34]

$$\mathbf{W}^{\ell} = \mathbf{S}_w\mathbf{W}^{\ell+1} \tag{4.12}$$

where $\mathbf{S}_w = \left\{\frac{5}{256}, \frac{20}{256}, \frac{1}{256}, \frac{-96}{256}, \frac{-70}{256}, \frac{280}{256}, \frac{-70}{256}, \frac{-96}{256}, \frac{1}{256}, \frac{20}{256}, \frac{5}{256}\right\}$. Note that $\mathbf{S}_w$ has been determined so that the basis has certain orthogonality properties. These relations enable conversion of coarse resolution B-spines and wavelets to fine resolution B-spines given the relation in (4.8)

$$\begin{aligned} C &= \mathbf{N}^0\mathbf{x}^0 + \sum_{m=0}^{\ell-1} \mathbf{W}^m\mathbf{d}^m \\ &= \mathbf{N}^{\ell-1}\mathbf{x}^{\ell-1} + \mathbf{W}^{\ell-1}\mathbf{d}^{\ell-1} \\ &= \mathbf{N}^{\ell}\mathbf{x}^{\ell} \end{aligned} \tag{4.13}$$

where $\mathbf{x}^{\ell+1} = \mathbf{S}\mathbf{x}^{\ell} + \mathbf{S}_w\mathbf{d}^{\ell}$. More importantly, the following relations [34] can be used to project data from fine to coarse resolutions;

$$\mathbf{x}^{\ell} = \mathbf{R}\mathbf{x}^{\ell+1} \tag{4.14}$$

$$\mathbf{d}^{\ell} = \mathbf{R}_w\mathbf{x}^{\ell+1} \tag{4.15}$$

with $\mathbf{R} = \left\{\frac{-5}{256}, \frac{20}{256}, \frac{-1}{256}, \frac{-96}{256}, \frac{70}{256}, \frac{280}{256}, \frac{70}{256}, \frac{-96}{256}, \frac{-1}{256}, \frac{20}{256}, \frac{-5}{256}\right\}$ and $\mathbf{R}_w = \left\{\frac{1}{8}, \frac{-1}{2}, \frac{3}{4}, \frac{-1}{2}, \frac{1}{8}\right\}$. These relations can be expressed using local operations similar to subdivision masks. However wavelets are not a good representation for multiresolution editing as the details

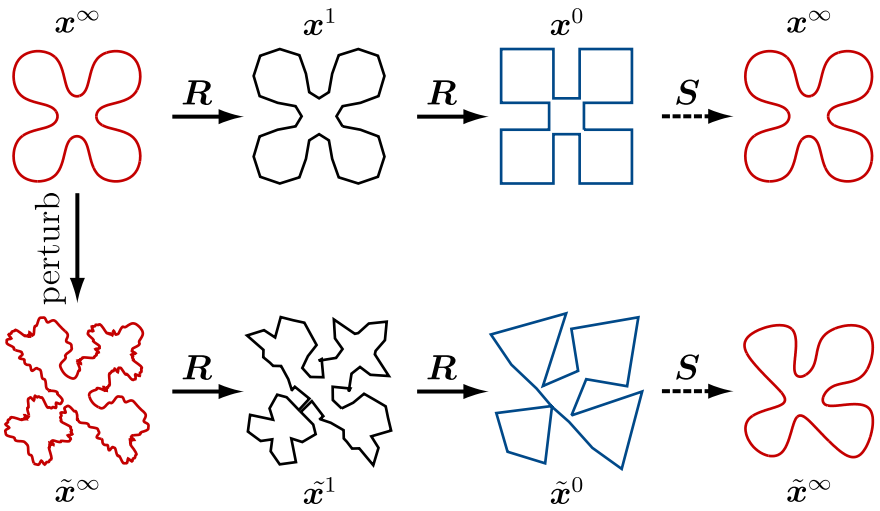


Figure 4.6.: Coarsening with B-spine wavelets. The limit curve previously obtained with subdivision refinement, see Figure 2.6, is coarsened (4.14). Top row shows the coarsening of the unperturbed limit surface and the bottom row shows the coarsening of a perturbed limit surface. For a vertex with index i the perturbation is of the form $\tilde{\boldsymbol{x}}_i{}^\infty = \boldsymbol{x}_i{}^\infty + \sum_j \boldsymbol{n}(\boldsymbol{x}_i) a_j \sin(f_j \boldsymbol{x}_i \cdot \boldsymbol{e}_0)$, where $\boldsymbol{n}$ is the normal to the curve, $\boldsymbol{e}_0$ is the basis vector $(1,0,0)^\mathsf{T}$, a is the amplitude and f is the angular frequency.

tend to behave in undesirable ways during editing [58, 113]. Additionally, developing wavelets schemes for arbitrary connectivity surface meshes is not straightforward.

In Figure 4.6 the functioning of B-spine wavelet coarsening operator is illustrated reconsidering the one-dimensional subdivision refinement example previously introduced in Figure 2.6. The coarsening of two limit curves is investigated. On the top row of Figure 4.6 the limit curve previously obtained via subdivision refinement in Figure 2.6 is successively coarsened (4.14) until the original control polygon is recovered. Note that this is possible due to the identity property (4.10). On the bottom row the coarsening of a perturbed limit surface is shown. As can be seen the coarsening operation successively removes the high-frequency oscillations from the geometry while satisfying the identity requirement (4.10). The resulting control polygon represents a limit surface which is a visually faithful smooth representation of the perturbed original curve.

Quasi-interpolation

Quasi-interpolation [38, 39] methods enable the construction of a spline surface to approximate a given function based on local operations. A quasi-interpolation method

for fitting subdivision surfaces to arbitrary surfaces is available in [88]. Assume $\mathbf{p}$ are sample points that need to be fitted using a subdivision mesh. Let $\mathbf{x}$ denote the control points of the subdivision mesh, the objective is to find $\mathbf{x}$ such that $\mathbf{p} = \mathbf{L}_0\mathbf{x}$, where $\mathbf{L}_0$ is the limit mask (A.3).

$$\mathbf{x} = \mathbf{L}_0^{-1}\mathbf{p} \tag{4.16}$$

An approximate inverse of the limit mask $\mathbf{L}_0$ can be obtained using a Neumann series;

$$\begin{aligned}\mathbf{L}_0^{-1} &= \sum_{i=0}^{\infty}(\mathbf{I} - \mathbf{L}_0)^i \\ &= \mathbf{I} + (\mathbf{I} - \mathbf{L}_0) + (\mathbf{I} - \mathbf{L}_0)^2 + \dots \\ &\approx 2\mathbf{I} - \mathbf{L}_0\end{aligned}$$

The quasi-interpolation operator $2\mathbf{I} - \mathbf{L}_0$ can be adapted to provide the required coarsening operator by fitting the control points $\mathbf{x}^\ell$ to the limit surface $\mathbf{p} = \mathbf{L}_0\mathbf{x}^{\ell+1}$

$$\mathbf{x}^\ell = (2\mathbf{I} - \mathbf{L}_0)\mathbf{p} \tag{4.17}$$

giving the coarsening operator $(2\mathbf{I} - \mathbf{L}_0)\mathbf{L}_0$ which can be expressed as local operations similar to reverse subdivision. However, the accuracy of the quasi-interpolation operator is comparable to least squares fitting except at extraordinary vertices where the error increases with the valence of the vertex [88].

Least squares fit

A common method for fitting meshes to arbitrary data is least squares fitting [71, 70]. The distance energy between the two geometry resolutions ℓ and $\ell + 1$ can be expressed by the following functional;

$$E_{dist} = \int_\Omega \|\mathbf{N}^{\ell+1}\mathbf{x}^{\ell+1} - \mathbf{N}^\ell\mathbf{x}^l\|^2 d\Omega \tag{4.18}$$

The coarse resolution control points $\mathbf{x}^\ell$ can be found by minimising this functional. However expressing the energy in a continuous sense as above requires evaluating the basis functions $\mathbf{N}^\ell$ and $\mathbf{N}^{\ell+1}$ at the quadrature points, a computationally expensive

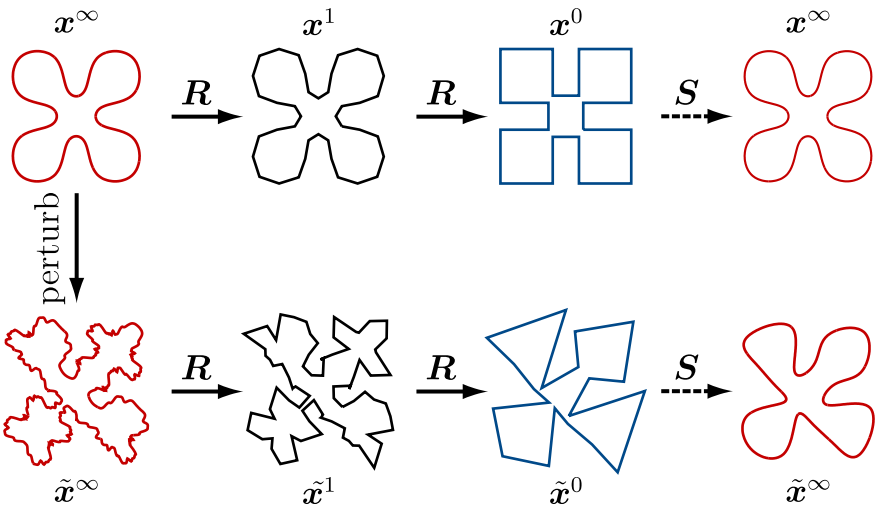


Figure 4.7.: Coarsening using least squares fitting. In (4.20), cubic subdivision is used to create $\mathbf{S}$. See Figure 4.6 for problem description.

process. Alternatively the distance energy can be expressed in a discrete form

$$E_{dist} = \|\mathbf{x}^{\ell+1} - \mathbf{S}\mathbf{x}^{\ell}\|^2 \tag{4.19}$$

The coarse resolution control points can be obtained by minimising the above

$$\mathbf{x}^{\ell} = \underset{\mathbf{x}^{\ell}}{\operatorname{argmin}} \|\mathbf{x}^{\ell+1} - \mathbf{S}\mathbf{x}^{\ell}\|^2 \tag{4.20a}$$

$$\mathbf{S}^{\mathsf{T}}\mathbf{S}\mathbf{x}^{\ell} = \mathbf{S}^{\mathsf{T}}\mathbf{x}^{\ell+1} \tag{4.20b}$$

$$\mathbf{x}^{\ell} = (\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathsf{T}}\mathbf{x}^{\ell+1} \tag{4.20c}$$

The desired coarsening operator (1.4) is now given by $\mathbf{R} = (\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathsf{T}}$. Note that since subdivision is a local operation, the $\mathbf{S}^{\mathsf{T}}\mathbf{S}$ term in (4.20c) can be assembled locally.

Figure 4.7 shows the functionality of the least squares coarsening operator with the same example coarsened with B-spine wavelets (Figure 4.6). Properties of the wavelet operator such as preserving identity and obtaining a smooth coarse resolution while eliminating only high frequency noise are recovered by the least square coarsening operator. The coarsening process (4.20) for any subdivision scheme, such as Loop and Catmull-Clark, can be established. Additionally, any special features in the subdivision surface, such as creases, are naturally recovered by the coarsening process (4.20) by design.

4.3.4. Integrated geometric design, finite element analysis and optimisation

With the coarsening operators discussed in the previous section, all the components of the multiresolution shape optimisation algorithm are in place. This description of Algorithm 1 should be read in conjunction with Figure 1.5. For simplicity of the algorithm, it is assumed that the discretised optimisation problem is solved with a steepest descent algorithm and no additional constraints (volume, area etc) are present.

Let $\ell_o = 0$ and ℓ_c denote the coarse resolution containing the design variables and the fine resolution containing the computational model, respectively. The fine resolution ℓ_c is user given and has to be large enough such that the accuracy of the numerical solution is sufficient for practical purposes. The initial geometry is denoted by $\mathbf{x}^{\ell_o}$ in this context. Optimisation of geometry is done in an incremental manner; design variables in resolution ℓ_o are first optimised before the optimisation level is incremented by one, i.e. $\ell_o \leftarrow (\ell_o + 1)$.

During each optimisation step, the model is subdivided $(\ell_c - \ell_o)$ times until the computation mesh $\mathbf{x}^{\ell_c}$ is obtained. In the present work, three different subdivision schemes are used. Uniform cubic subdivision (Figure 2.7) rules are used for optimising 2D geometry while Catmull-Clark (Figure 2.8) and Loop (Figure 2.9) are used in the 3D case. The objective function (4.3a) and shape derivatives (4.4) are evaluated using this fine resolution model. The shape derivative is interpolated at the fine resolution nodes and projected to the coarse resolution using (4.20). Note that, depending on the subdivision scheme used, the coarsening operator is different each time. The optimisation problem is solved in the coarse resolution and the design variable updated using a suitable mathematical programming method. In the actual implementation the Method of Moving Asymptotes (MMA) proposed by Svanberg [123, 124] as implemented in the nlopt library [76] is used to this end. This changes in particular Step 9 in the Algorithm 1. As to be expected, using a more sophisticated optimisation algorithm than steepest descent significantly reduces the number of optimisation iterations. If any additional constraints such as volume or area constraints are present, they must be evaluated and their derivatives computed after step 6. The derivative of the constraints also need to be projected to the coarse resolution in step 8.

Algorithm 1 Multiresolution optimisation with subdivision

// initiate optimisation level ℓ_o
$\ell_o = 0$
// iterate until optimisation level is equal to the analysis level ℓ_c
while $\ell_o < \ell_c$ **do**
  // update vertex coordinates $\mathbf{x}^{\ell_o}$ while objective function decreases
  repeat
    // subdivide geometry model on level ℓ_o up to analysis level ℓ_c
    $\mathbf{x}^{\ell_c} = \mathbf{S}^{(\ell_c - \ell_o)}\mathbf{x}^{\ell_o}$
    // compute objective function c and shape derivative γ
    $c = \mathcal{J}(\boldsymbol{x}^{\ell_c}, \boldsymbol{u}(\boldsymbol{x}^{\ell_c})), \quad \boldsymbol{x}^{\ell_c} \in \mathbf{x}^{\ell_c}$
    $\gamma = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tau}(\boldsymbol{x}^{\ell_c}, \boldsymbol{u}(\boldsymbol{x}^{\ell_c}))$
    // interpolate shape derivative at fine resolution nodes
    $\gamma \mapsto \mathbf{g}^{\ell_c}$
    // project shape derivative to optimisation level
    $\mathbf{g}^{\ell_o} = \mathbf{R}^{(\ell_c - \ell_o)}\mathbf{g}^{\ell_c}$
    // update vertex coordinates of the optimisation model
    $\boldsymbol{x}_i^{\ell_o} \leftarrow (\boldsymbol{x}_i^{\ell_o} + \alpha \boldsymbol{g}_i^{\ell_o}), \quad \alpha < 0,\ \boldsymbol{x}_i^{\ell_o} \in \mathbf{x}_o^{\ell},\ \boldsymbol{g}_i^{\ell_o} \in \mathbf{g}_o^{\ell}$
  until $c < c_{\text{prev}}$
  // increment optimisation level
  $\ell_o \leftarrow (\ell_o + 1)$
  $\mathbf{x}^{\ell_o} \leftarrow \mathbf{S}\mathbf{x}^{\ell_o}$
end while

Figure 4.8 shows an application of the multiresolution framework using subdivision. The depicted example is a shell optimisation problem where the shape of an architectural roof is optimised for a uniform pressure load. The example will be covered in detail later in Chapter 5. In this example, no additional geometry changes are made in the design stage. If any such additional design features are present, they must be stored as details using (4.9).

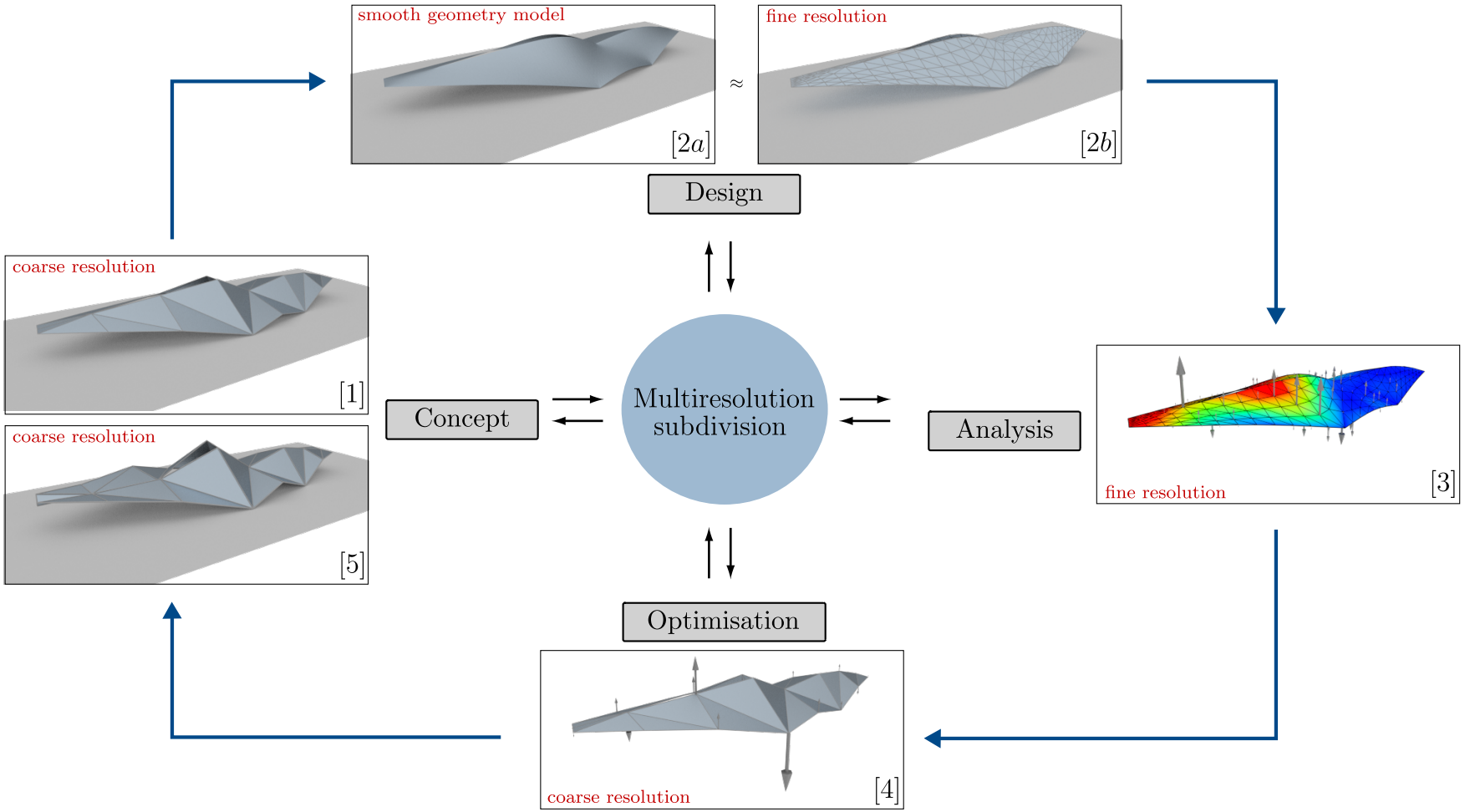
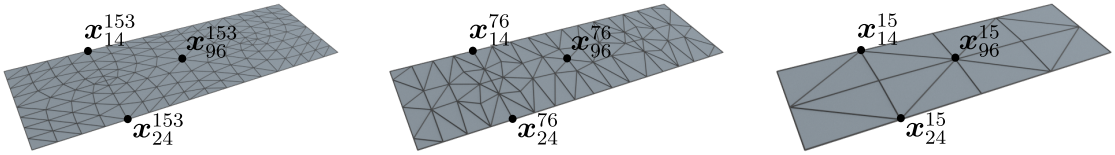


Figure 4.8.: Multiresolution subdivision framework (c.f. Figure 1.6). Concept stage creates a coarse control mesh on which the design variables are defined. This has a corresponding smooth geometry model which represents the current design. The analysis stage uses a suitable fine resolution mesh that approximates the geometry model. The vectors shown in the optimisation and analysis stages indicate the discretised shape derivatives and the colour contours in the latter represent displacement.



(a) Initial mesh $\mathbf{x}^{153}$ (b) Coarse resolution $\mathbf{x}^{76}$ (c) Coarse resolution $\mathbf{x}^{15}$

Figure 4.9.: Multiple resolutions of flat rectangular plate (0.3×1). The Initial mesh $\mathbf{x}^{153}$ is coarsened with 77 and 138 halfedge collapses to obtain coarse resolutions $\mathbf{x}^{76}$ and $\mathbf{x}^{15}$ respectively. Note that binary oracles have been used to obtain relatively uniform elements in each of the coarse resolutions and to preserve nodes with indices $14, 24$ and 96.

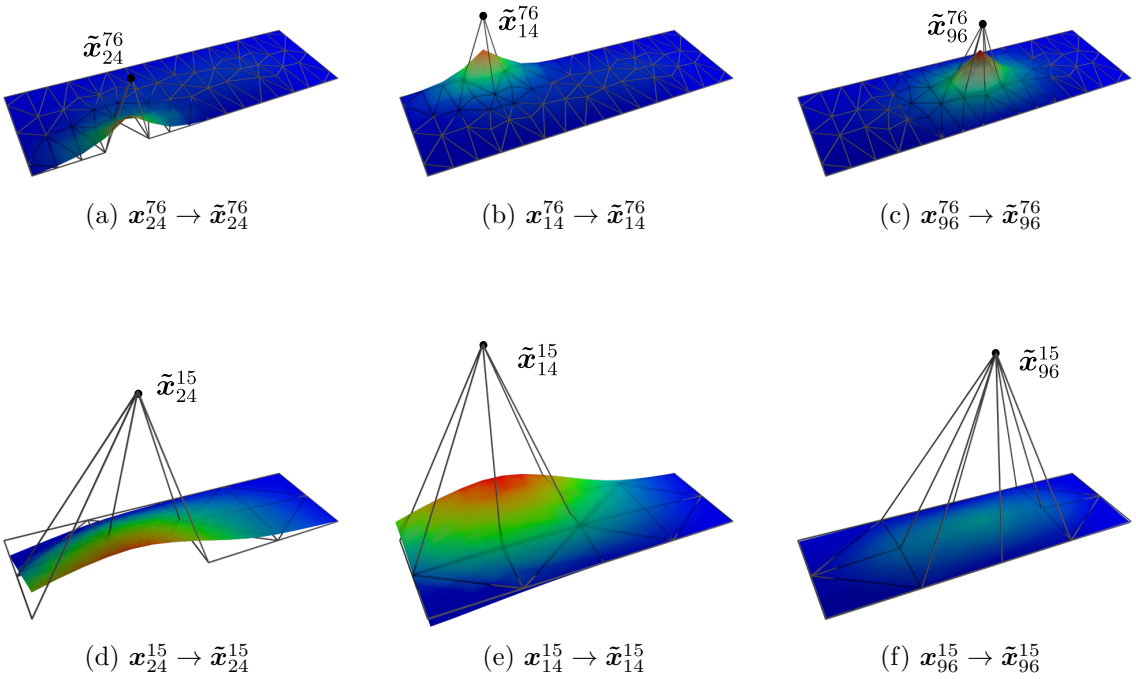
4.4. Bottom-up multiresolution framework

4.4.1. Basis function analogy

Multiresolution representations based on mesh decimation are not naturally suited to the structural concept of considering a surface in space to be a linear combination of scalar-valued basis functions [82] such as shape functions in finite elements. However, a graphical representation of the underlying basis functions can be obtained. Figure 4.9 shows multiple coarse resolutions of a flat plate. Several selected nodes in each resolution are perturbed along the normal direction and the details restored as shown in Figure 4.10. The deformed fine resolution mesh can be identified as the corresponding basis function for the perturbed node.

4.4.2. Uniform geometry editing

In a subdivision setting, the limit position of a node in a deformed control mesh is determined using limit masks (A.3). In progressive meshes, the equivalent to a limit surface is the fine resolution mesh. Due to the Laplacian smoothing of the one-ring for each decimated node (Section 3.3.2), the limit position (position in the fine resolution mesh with details restored) of each perturbed node depends on the decimation process. This is reflected by the different maximum values for each of the shape functions in Figure 4.10. As a consequence, perturbing nodes of a uniform coarse resolution mesh



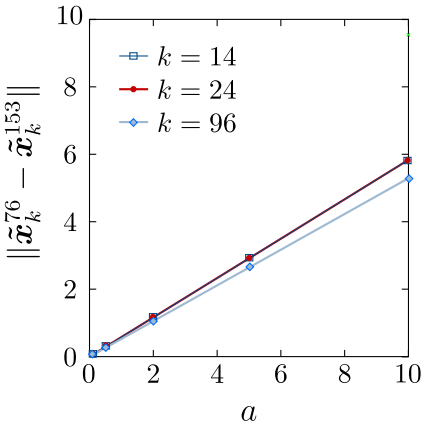
(a) $\boldsymbol{x}_{24}^{76} \to \tilde{\boldsymbol{x}}_{24}^{76}$ (b) $\boldsymbol{x}_{14}^{76} \to \tilde{\boldsymbol{x}}_{14}^{76}$ (c) $\boldsymbol{x}_{96}^{76} \to \tilde{\boldsymbol{x}}_{96}^{76}$

(d) $\boldsymbol{x}_{24}^{15} \to \tilde{\boldsymbol{x}}_{24}^{15}$ (e) $\boldsymbol{x}_{14}^{15} \to \tilde{\boldsymbol{x}}_{14}^{15}$ (f) $\boldsymbol{x}_{96}^{15} \to \tilde{\boldsymbol{x}}_{96}^{15}$

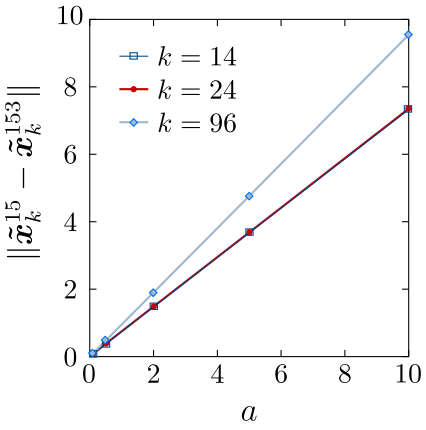
Figure 4.10.: Progressive mesh shape functions for example in Figure 4.9. Nodes in coarse resolution ℓ are perturbed as $\tilde{\boldsymbol{x}}_i^\ell = \boldsymbol{x}_i^\ell + a\boldsymbol{n}$, where $i \in \{14, 24, 96\}$ and $\boldsymbol{n}$ is the unit normal to the plate. The amplitude is set to $a = 0.25$ for coarse resolution $\ell = 76$ (Figure 4.9b) and $a = 0.5$ for coarse resolution $\ell = 15$ (Figure 4.9c). In each figure, the wireframe indicates the deformed coarse resolution mesh and the colour contours indicate elevation of the deformed fine resolution mesh with respect to the plate surface.

may result in non-uniform surface deformations which is not desirable for multiresolution geometry editing.

Use of constant weights in Laplacian smoothing (3.8a) imply linear response to a perturbation of a coarse resolution node. Let $\mathbf{x}^\ell$ denote the coarse resolution mesh obtained after $(n-\ell)$ halfedge collapses of some original fine resolution mesh $\mathbf{x}^n$. A node $\boldsymbol{x}_k^\ell \in \mathbf{x}^\ell$ is now perturbed $\boldsymbol{x}_k^\ell \to \boldsymbol{x}_k^\ell + a\boldsymbol{n}$ along the unit normal $\boldsymbol{n}$. Note the magnitude of perturbation is given by a. The details are now restored with the position of the perturbed node in the fine resolution mesh denoted by $\tilde{\mathbf{x}}_k^n$. The shrinking due to Laplacian smoothing is now represented by the vector $\tilde{\mathbf{x}}_k^\ell - \tilde{\mathbf{x}}_k^n$. Figure 4.11 plots shrinking magnitude $\|\tilde{\mathbf{x}}_k^\ell - \tilde{\mathbf{x}}_k^n\|$ for different perturbation magnitudes a in Figure 4.10 showing a linear relationship between the two. This is useful, for example, enforcing prescribed deformations on a mesh as shown in Figure 4.12. Note that the linear behaviour is lost if non-uniform weights, such as the inverse of edge length [125], or local frames (Section 3.3.2) are used.



(a) Shape functions shown in Figures 4.10a, 4.10b and 4.10c with $\ell = 76$.



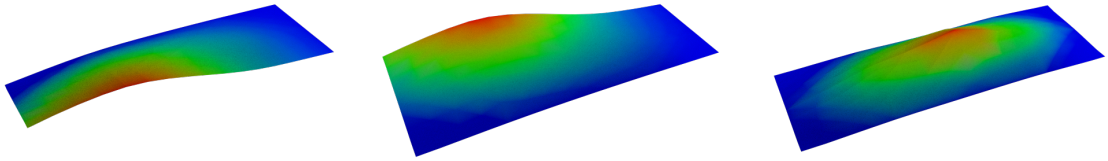
(b) Shape functions shown in Figures 4.10d, 4.10e and 4.10f with $\ell = 15$.

Figure 4.11.: Relationship between shrinking and perturbation for shape functions in Figure 4.10. Shrinking $\|\tilde{\mathbf{x}}_k^{\ell} - \tilde{\mathbf{x}}_k^{153}\|$ is plotted for different perturbation magnitudes a.

4.4.3. Coarsening of field data

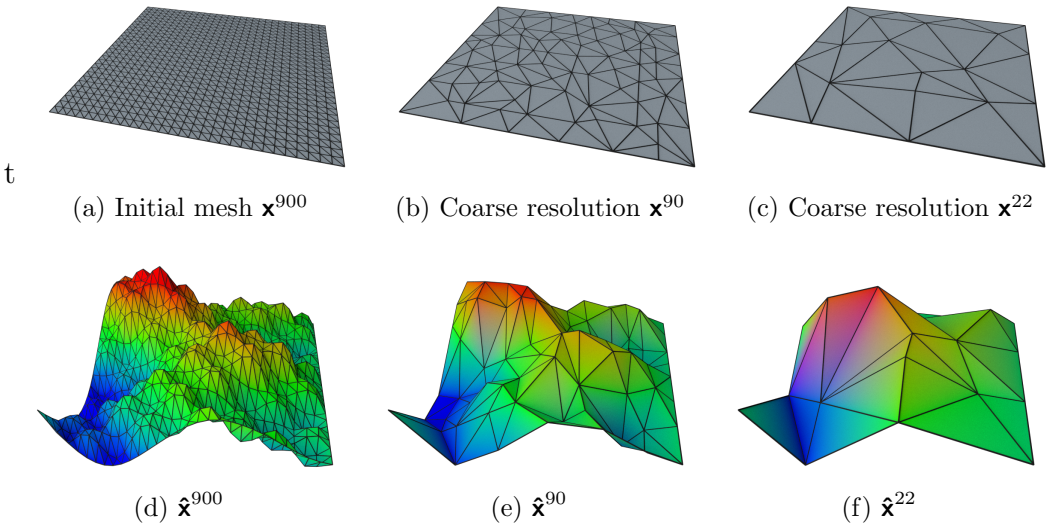
During multiresolution optimisation, the computed fine resolution design sensitivity needs to be projected to the coarse resolution. In the quadric based decimation method described in Section 3.2, coarse resolutions are obtained by removing nodes via halfedge collapses. This implies the coarse resolution nodes are simply subsets of the fine resolution and the coarsening operator is a sub-sampling operation.

Figure 4.13 shows an example where sub-sampling based coarsening of some field data associated to nodes is demonstrated. It is clear that correspondence between the original data field and the projected coarse resolution data field gradually deteriorates with coarsening. One remedy is to have a pre-smoothing step before computing details [59]. Essentially, the one-ring of the decimated vertex can be smoothed via a non-shrinking Laplacian based smoothing operation resulting in smoother coarse resolutions. This requires the original wiring diagram for the decomposition step (Figure 3.7a) to be modified as indicated in Figure 4.14. However, in practice such pre-smoothing can be avoided due to the incremental nature of the multiresolution framework, i.e. later stages of the optimisation process are done on much finer resolutions where the field data is adequately represented. Additionally, extreme coarse resolutions are not suitable for optimisation with progressive meshes. This is due to the large detail vectors present in such cases causing undesirable effects during detail restoration.



(a) $\boldsymbol{x}_{24}^{15} \rightarrow \tilde{\boldsymbol{x}}_{24}^{15}$ (b) $\boldsymbol{x}_{24}^{15} \rightarrow \tilde{\boldsymbol{x}}_{24}^{15}$ (c) $\boldsymbol{x}_{96}^{15} \rightarrow \tilde{\boldsymbol{x}}_{96}^{15}$

Figure 4.12.: Progressive mesh shape functions for uniform geometry editing. The perturbations corresponding to shape functions in Figures 4.10d, 4.10e and 4.10f have been scaled to obtain uniform geometry editing. The perturbations are of the form to $\tilde{\boldsymbol{x}}_i^{15} = \boldsymbol{x}_i^{15} + \hat{a}\boldsymbol{n}, \quad i \in \{14, 24, 96\}$ where $\boldsymbol{n}$ is the unit normal. The amplitude $\hat{a}$ for each node is multiplied by a factor $1/(1 - \rho_{\text{sf}})$ where ρ_{sf} is the gradient of the curve in Figure 4.11b.

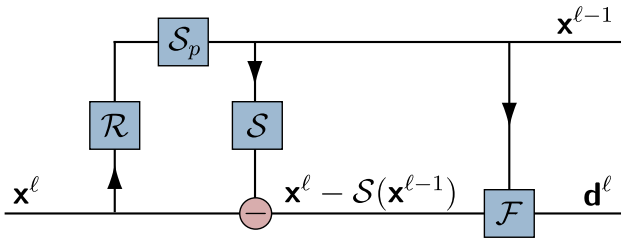


t

(a) Initial mesh $\mathbf{x}^{900}$ (b) Coarse resolution $\mathbf{x}^{90}$ (c) Coarse resolution $\mathbf{x}^{22}$

(d) $\hat{\mathbf{x}}^{900}$ (e) $\hat{\mathbf{x}}^{90}$ (f) $\hat{\mathbf{x}}^{22}$

Figure 4.13.: Coarsening of field data using sub-sampling. Nodes in the initial mesh $\mathbf{x}^{900}$ are perturbed using a scalar field f to $\tilde{\boldsymbol{x}}_i^{900} = \boldsymbol{x}_i^{900} + f(\boldsymbol{x}_i^{900})\boldsymbol{n}$ where $\boldsymbol{n}$ is the normal to the surface. Next the mesh is coarsened with 810 and 878 halfedge collapses to obtain coarse resolutions $\mathbf{x}^{90}$ and $\mathbf{x}^{22}$ respectively (top). In the bottom row, the colour contours and the elevation about the plate surface indicate the scalar field f.



$\mathcal{R}$: decimation $\mathcal{S}$: smoothing $\mathcal{F}$: local frame $\mathcal{S}_p$: pre-smoothing

Figure 4.14.: A pre-smoothing operation $\mathcal{S}_p$ can be added [59] to the original decomposition scheme in Figure 3.7a. The pre-smoothing ensures that each coarse resolution is a smoothed representation of the fine resolution data.

4.4.4. Integrated design, analysis and optimisation algorithm

This description of the bottom-up multiresolution optimisation Algorithm 2 using progressive meshes should be read in conjunction with Figure 1.9. Same settings as in the subdivision algorithm (Algorithm 1) are assumed where it is assumed that the discretised optimisation problem is solved with a steepest descent algorithm and no additional constraints (volume, area etc) are present.

Let $\mathbf{x}^{\ell_c}$ denote the fine resolution initial design used as the starting point for optimisation. The analysis module will use this same mesh, hence the resolution ℓ_c needs to be such that the accuracy of the numerical solution is sufficient for practical purposes. Assume the set $\mathcal{K} = \{\ell_1, \ell_2, \ldots, \ell_k\}, \quad \ell_k \leq \ell_c$ contains the sequence of increasing resolutions specified by the user as the different optimisation resolutions, i.e. design variables in resolution ℓ_1 are first optimised before they are moved to the next resolution ℓ_2 etc. In practice, such resolutions are obtained by decimating some percentage of nodes in the starting geometry mesh.

The progressive mesh hierarchy is built once for each optimisation resolution $\ell_o \in \mathcal{K}$. This involves performing $\ell_c - \ell_o$ halfedge collapses (Section 3.2) on the fine resolution mesh $\mathbf{x}^{\ell_c}$. The set of details $\mathcal{D}$ computed (Equations 3.13 and 3.14) during the decimation will be used throughout iterative optimisation of design variables in $\mathbf{x}^{\ell_o}$. The objective function (4.3a) and shape derivatives (4.4) are always evaluated using the fine resolution model $\mathbf{x}^{\ell_c}$. Due to the sub-sampling nature of the coarsening process (Section 4.4.3), the shape derivative γ only needs to be interpolated at fine resolution nodes that are not decimated during the coarsening $\mathbf{x}^{\ell_c} \rightarrow \mathbf{x}^{\ell_o}$. However if pre-smoothing is

present (Figure 4.14), γ needs to be discretised to all fine resolution nodes. Solving the optimisation problem in the coarse resolution follows the same order as Algorithm 1.

If uniform geometry modifications are required (Section 4.4.2), the relationship between shrinking and perturbation of shape functions needs to be pre-computed before step 6. This information can be used when updating the positions of the coarse resolution design variables in step 13.

Algorithm 2 Multiresolution optimisation with progressive meshes

// obtain sequence of optimisation resolutions
1: $\mathcal{K} = \{\ell_1, \ell_2, \ldots, \ell_k\}, \quad \ell_k \leq \ell_c$
// optimise each coarse resolution specified in $\mathcal{K}$
2: **for** $\ell = \ell_o \in \mathcal{K}$ **do**
// decimate mesh to coarse resolution ℓ*, collect and store set of details* $\mathcal{D}$
3: $\mathcal{D} = \{\mathbf{d}^{\ell_c}, \mathbf{d}^{\ell_c - 1}, \ldots, \mathbf{d}^{\ell_o - 1}\}$
4: $\mathbf{d}^r = \mathbf{x}^r - \mathcal{S}(\mathbf{x}^{r-1}), \quad \mathbf{d}^r \in \mathcal{D}$
5: $\mathbf{x}^{\ell_c} \xrightarrow{\text{hcol}^1} \mathbf{x}^{\ell_c - 1} \xrightarrow{\text{hcol}^2} \mathbf{x}^{\ell_c - 2} \ \ldots \ \xrightarrow{\text{hcol}^{\ell_c - \ell_o}} \mathbf{x}^{\ell_o}$
// optimise design variables $\mathbf{x}^{\ell_0}$ *while objective function decreases*
6: **repeat**
// restore details up to fine resolution ℓ_c
7: $\mathbf{x}^r = \mathbf{d}^r + \mathcal{S}(\mathbf{x}^\ell) \quad \mathbf{d}^r \in \mathcal{D}$
8: $\mathbf{x}^{\ell_c} \xleftarrow{\text{vsplit}^{\ell_c - \ell_o}} \ldots \xleftarrow{\text{vsplit}^2} \mathbf{x}^{\ell_o - 1} \xleftarrow{\text{vsplit}^1} \mathbf{x}^{\ell_o}$
// compute objective function c *and shape derivative* γ
9: $c = \mathcal{J}(\boldsymbol{x}^{\ell_c}, \boldsymbol{u}(\boldsymbol{x}^{\ell_c})), \quad \boldsymbol{x}^{\ell_c} \in \mathbf{x}^{\ell_c}$
10: $\gamma = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tau}(\boldsymbol{x}^{\ell_c}, \boldsymbol{u}(\boldsymbol{x}^{\ell_c}))$
// interpolate shape derivative at fine resolution nodes
11: $\gamma \mapsto \mathbf{g}^{\ell_c}$
// decimate mesh to resolution ℓ *(not required to recompute details)*
12: $\mathbf{x}^{\ell_c} \xrightarrow{\text{hcol}^1} \mathbf{x}^{\ell_c - 1} \xrightarrow{\text{hcol}^2} \mathbf{x}^{\ell_c - 2} \ \ldots \ \xrightarrow{\text{hcol}^{\ell_c - \ell_o}} \mathbf{x}^{\ell_o}$
// update design variables in coarse resolution
13: $\boldsymbol{x}_i^{\ell_o} \leftarrow (\boldsymbol{x}_i^{\ell_o} + \alpha \boldsymbol{g}_i^{\ell_c}), \quad \alpha < 0, \ \boldsymbol{x}_i^{\ell_o} \in \mathbf{x}^{\ell_o}, \ \boldsymbol{g}_i^{\ell_c} \in \mathbf{g}^{\ell_c}$
14: **until** $c < c_{\text{prev}}$
15: **end for**

Note that both Algorithms 1 and 2 essentially follow the same workflow. The main for loop is for optimising the design variables in each optimisation level ℓ_o. For each such level, the cost function is repeatedly computed and the positions of the design variables are iteratively updated. The main difference between the two algorithms being the presence of details in Algorithm 2 and the difference in multiresolution data transfer operations. In Algorithm 1, the subdivision matrix $\mathbf{S}$ and the coarsening operator $\mathbf{R}$

are used for data transfer while Algorithm 2 uses detail restoration via the smoothing operator $\mathcal{S}$ and mesh decimation.

Figure 4.15 shows an application of the multiresolution framework using progressive. The same example as in Figure 4.8 is used but using a more detailed fine resolution mesh with arbitrary connectivity as the starting geometry.

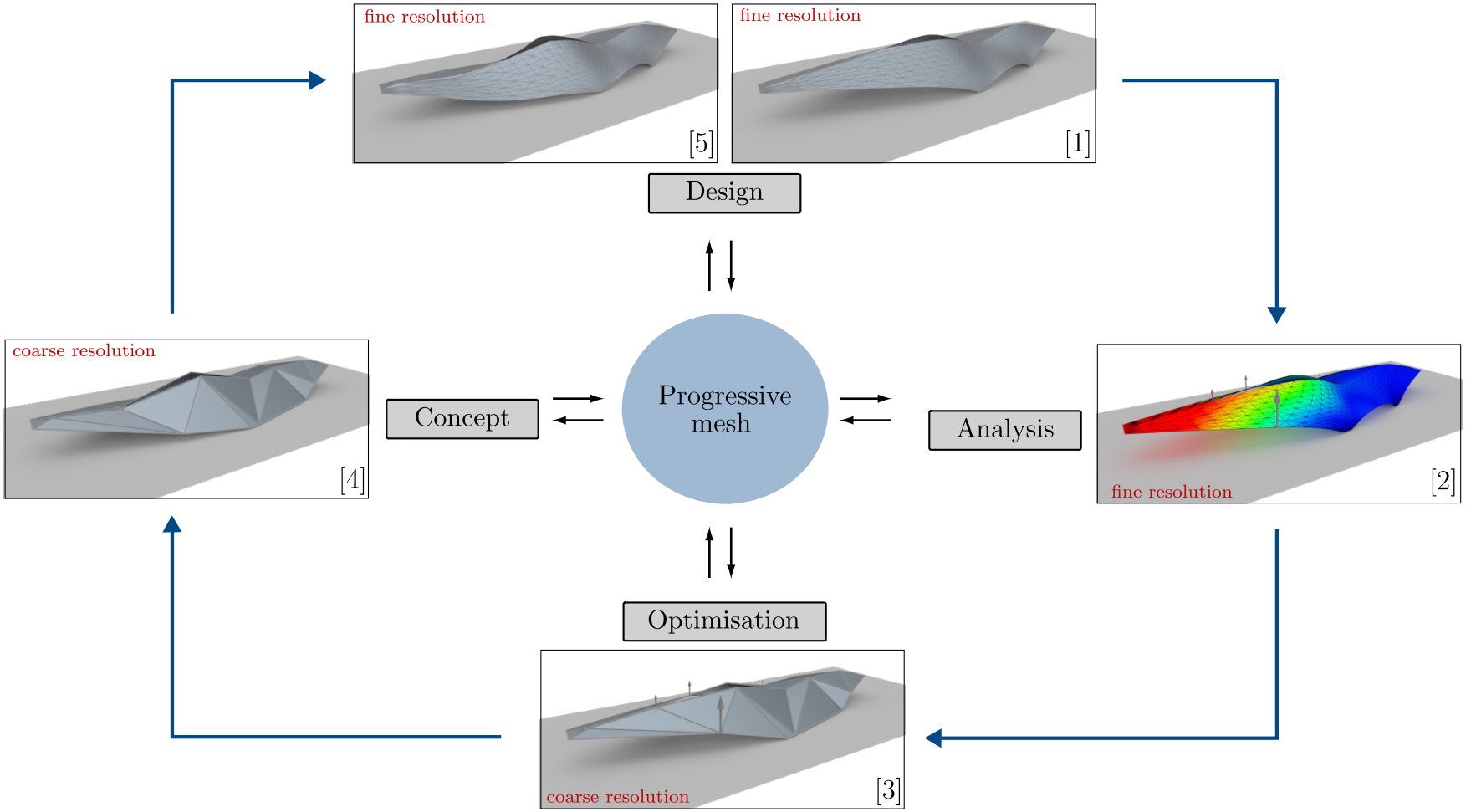


Figure 4.15.: Multiresolution progressive mesh framework (c.f. Figure 4.8). The starting geometry is a given fine resolution design model which is directly used in the analysis stage. Mesh decimation is used to obtain a coarse resolution model for optimisation. Note that the latter resembles a conceptual geometric model of the given design model. The design variables in this coarse resolution model are optimised and the resulting coarse level changes are transferred to the fine resolution via detail restoration. The vectors shown in the optimisation and analysis stages indicate the discretised shape derivatives and the colour contours in the latter represent displacement.

5. Kirchhoff-Love shell optimisation

Kirchhoff-Love theory offers an ideal formulation for parameter-free shape optimisation of thin shells since it is displacement based and does not require rotational degrees of freedom. In this setting, the energy functional contains the second order derivatives of displacement. This leads to the C^1-continuity requirement for the finite element basis functions, which need to have square integrable second order derivatives. Subdivision based finite element methods offer an elegant solution given the underlying surfaces are C^2 at all places except at a few extraordinary points where the surface is only C^1. Similar C^1 methods are available using NURBS basis functions [73, 78, 77] and can be used for optimisation in combination with a NURBS based multiresolution framework.

Multiresolution shape optimisation of thin shells using Kirchhoff-Love formulation is presented in this chapter. The subdivision based isogeometric method developed by Cirak et al. [33, 32, 30] is used for solving the shell equations and computing the design sensitivities which are derived analytically.

The thin shell problem is formulated on the fine resolution mesh with subdivision shell discretisation. In the subdivision multiresolution framework (Section 4.3) the fine resolution is obtained by refining the given coarse resolution whereas in the progressive mesh framework (Section 4.4), the given mesh is directly used as the finite element mesh.

5.1. Review of thin shell mechanics

This section contains a brief review of thin shell kinematics and discretisation of the energy functional within the Kirchhoff-Love theory of thin shells. See Ciarlet [28], Cirak [33] for more details.

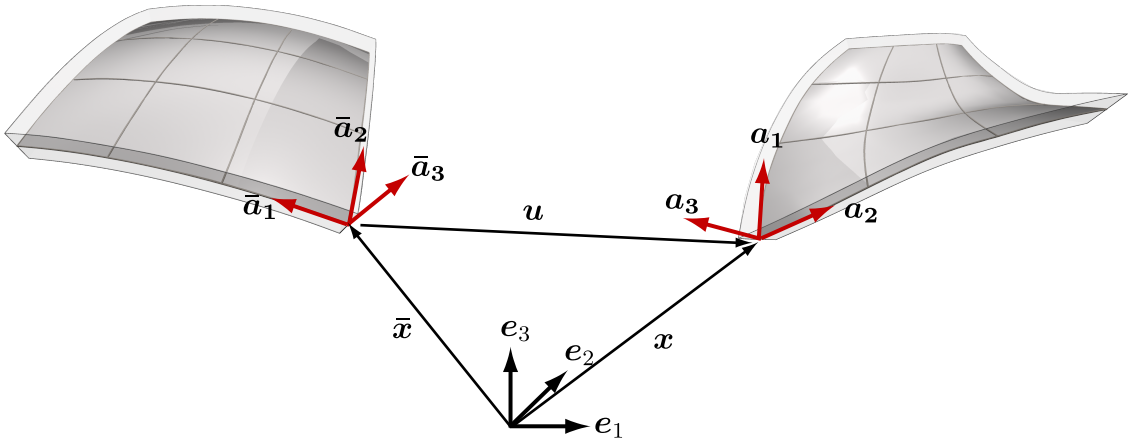


Figure 5.1.: Shell kinematic description. Reference configuration is shown on left and the current configuration is shown on right.

5.1.1. Kinematics

The Kirchhoff-Love theory of thin shells assumes that normals to the shell mid-surface remain normal during deformation implying no out-of-plane shear deformations are present. As a result, the behaviour of the shell can be described by using only its mid surface. Let $\bar{\boldsymbol{r}}$ and $\boldsymbol{r}$ denote the position vectors of a material point in the reference and deformed configurations, respectively. The curvilinear coordinate system $\{\theta^1, \theta^2, \theta^3\}$ can be used for parameterising a shell of thickness t as follows;

$$\bar{\boldsymbol{r}}(\theta^1, \theta^2, \theta^3) = \bar{\boldsymbol{x}}(\theta^1, \theta^2) + \theta^3 \bar{\boldsymbol{a}}_{\boldsymbol{3}}(\theta^1, \theta^2), \qquad -t/2 \leq \theta^3 \leq t/2 \tag{5.1a}$$

$$\boldsymbol{r}(\theta^1, \theta^2, \theta^3) = \boldsymbol{x}(\theta^1, \theta^2) + \theta^3 \boldsymbol{a}_{\boldsymbol{3}}(\theta^1, \theta^2), \qquad -t/2 \leq \theta^3 \leq t/2 \tag{5.1b}$$

where $\bar{\boldsymbol{x}}$ and $\boldsymbol{x}$ are the position vectors of the reference and deformed shell mid surface (Figure 5.1). The unit normal vectors to the shell mid-surface denoted by $\bar{\boldsymbol{a}}_{\boldsymbol{3}}$ and $\boldsymbol{a}_3$ in each configuration are given by

$$\bar{\boldsymbol{a}}_{\boldsymbol{3}} = \frac{\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2}{|\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2|}, \quad \boldsymbol{a}_3 = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} \tag{5.2}$$

using the covariant basis vectors of the mid surface in each configuration;

$$\bar{\boldsymbol{a}}_\alpha = \frac{\partial \bar{\boldsymbol{x}}(\theta^1, \theta^2)}{\partial \theta^\alpha}, \quad \boldsymbol{a}_\alpha = \frac{\partial \boldsymbol{x}(\theta^1, \theta^2)}{\partial \theta^\alpha} \tag{5.3}$$

The corresponding contravariant basis is given by $\bar{\boldsymbol{a}}^i \cdot \bar{\boldsymbol{a}}_j = \delta^i_j$ where δ^i_j is the Kronecker delta. Note that in the above and henceforth, the Greek indices can have the values $\alpha, \beta, \gamma, \delta \in \{1, 2\}$ and Latin indices can have the values $i, j \in \{1, 2, 3\}$.

Green-Lagrange strain tensor

According to the Kirchhoff-Love hypothesis the mid-surface normal $\boldsymbol{a}_3$ is not an independent vector. In this setting, the Green-Lagrange strain tensor which represents the deformation of the shell body can be expressed as follows (neglecting quadratic terms in θ^3)

$$\boldsymbol{E} = \boldsymbol{\alpha} + \theta^3 \boldsymbol{\beta} \tag{5.4}$$

where the tensors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are related to in-plane deformations of the shell mid surface and curvature change, respectively. The components of these tensors are given by

$$\alpha_{ij} = \frac{1}{2}(\boldsymbol{a}_i.\boldsymbol{a}_j - \bar{\boldsymbol{a}}_{\boldsymbol{i}} \cdot \bar{\boldsymbol{a}}_{\boldsymbol{j}}) \tag{5.5a}$$

$$\beta_{ij} = \boldsymbol{a}_i \cdot \boldsymbol{a}_{3,j} - \bar{\boldsymbol{a}}_{\boldsymbol{i}} \cdot \bar{\boldsymbol{a}}_{\boldsymbol{3,j}} \tag{5.5b}$$

Given $\boldsymbol{a}_3 \cdot \boldsymbol{a}_3 = \bar{\boldsymbol{a}}_{\boldsymbol{3}} \cdot \bar{\boldsymbol{a}}_{\boldsymbol{3}} = 1$ by definition and $\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_3 = \bar{\boldsymbol{a}}_{\boldsymbol{\alpha}} \cdot \bar{\boldsymbol{a}}_{\boldsymbol{3}}$ due to the Kirchhoff-Love assumption, the non-zero components of (5.5a) and (5.5b) reduce to;

$$\alpha_{\alpha\beta} = \frac{1}{2}(\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta - \bar{\boldsymbol{a}}_{\boldsymbol{\alpha}} \cdot \bar{\boldsymbol{a}}_{\boldsymbol{\beta}}) \tag{5.6a}$$

$$\beta_{\alpha\beta} = \bar{\boldsymbol{a}}_{\boldsymbol{\alpha,\beta}}.\bar{\boldsymbol{a}}_{\boldsymbol{3}} - \boldsymbol{a}_{\boldsymbol{\alpha,\beta}}.\boldsymbol{a}_3 \tag{5.6b}$$

It is evident that the deformation state of the shell is completely defined by the tensors on the mid-surface owing to the Kirchhoff-Love assumptions. Let $\boldsymbol{u}$ denote the displacement of the mid surface of the shell given by

$$\boldsymbol{u}(\theta^1, \theta^2) = \boldsymbol{x}(\theta^1, \theta^2) - \bar{\boldsymbol{x}}(\theta^1, \theta^2) \tag{5.7}$$

Assuming that deformation of the shells is small, geometrically linear kinematic equations can be adopted for linear thin shell problems. Linearisation of (5.6a) and (5.6b)

to first order of $\boldsymbol{u}$ results in

$$\alpha_{\alpha\beta} = \frac{1}{2}(\boldsymbol{u}_{,\alpha}.\bar{\boldsymbol{a}}_{\boldsymbol{\beta}} - \bar{\boldsymbol{a}}_{\boldsymbol{\alpha}} \cdot \boldsymbol{u}_{,\beta}) \tag{5.8a}$$

$$\begin{aligned}\beta_{\alpha\beta} = &-\boldsymbol{u}_{,\alpha\beta} \cdot \bar{\boldsymbol{a}}_{\boldsymbol{3}}+ \\ &\frac{1}{|\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2|}[\boldsymbol{u}_{,1} \cdot (\bar{\boldsymbol{a}}_{\alpha,\beta} \times \bar{\boldsymbol{a}}_2) + \boldsymbol{u}_{,2} \cdot (\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_{\alpha,\beta})]+ \\ &\frac{\boldsymbol{a}_3 \cdot \boldsymbol{a}_{\alpha,\beta}}{|\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2|}[\boldsymbol{u}_{,1} \cdot (\bar{\boldsymbol{a}}_2 \times \bar{\boldsymbol{a}}_3) + \boldsymbol{u}_{,2} \cdot (\bar{\boldsymbol{a}}_3 \times \bar{\boldsymbol{a}}_1)]\end{aligned} \tag{5.8b}$$

After the above linearisation, the reference and current domains are the same within first order.

5.1.2. Discretisation of the energy functional

In the present work, discretisation of shell energy follows the formulation discussed in Cirak et al. [33]. The total potential energy of the shell body reads

$$\begin{aligned}\Pi(\boldsymbol{u}) =&\Pi^{int}(\boldsymbol{u}) + \Pi^{ext}(\boldsymbol{u}) \\ =&\int_\Omega W^m(\boldsymbol{\alpha}) + W^b(\boldsymbol{\beta})\Omega + \Pi^{ext}(\boldsymbol{u})\end{aligned} \tag{5.9}$$

where Π^{ext} is the potential of the external forces and W^m, W^b are the membrane and bending energy densities, respectively. The potential of the external forces can be expressed as follows when the shell is subject to a system of external dead loads consisting of distributed loads $\boldsymbol{q}$ per unit area of mid-surface Ω and axial forces $\boldsymbol{N}$ per unit length of the boundary Γ;

$$\Pi^{ext}(\boldsymbol{u}) = -\int_\Omega \boldsymbol{q} \cdot \boldsymbol{u} d\Omega - \int_\Gamma \boldsymbol{N} \cdot \boldsymbol{u} d\Gamma \tag{5.10}$$

The membrane and bending energy densities can be expressed as;

$$W^m(\boldsymbol{\alpha}) =\frac{1}{2}\frac{Et}{1-\nu^2}H^{\alpha\beta\gamma\delta}\alpha_{\alpha\beta}\alpha_{\gamma\delta} \tag{5.11a}$$

$$W^b(\boldsymbol{\beta}) =\frac{1}{2}\frac{Et^3}{12(1-\nu^2)}H^{\alpha\beta\gamma\delta}\beta_{\alpha\beta}\beta_{\gamma\delta} \tag{5.11b}$$

where E is the Young's modulus, ν the Poisson's ratio and

$$H^{\alpha\beta\gamma\delta} = \nu\bar{a}^{\alpha\beta}\bar{a}^{\gamma\delta} + \frac{1}{2}(1-\nu)(\bar{a}^{\alpha\gamma}\bar{a}^{\beta\delta} + \bar{a}^{\alpha\delta}\bar{a}^{\beta\gamma}) \tag{5.12}$$

Note that $\bar{a}^{\alpha\beta} = \bar{\boldsymbol{a}}^{\alpha} \cdot \bar{\boldsymbol{a}}^{\beta}$ are the contravariant components of the metric tensor for the reference configuration. The membrane and bending stresses can be obtained by differentiation

$$n^{\alpha\beta} = \frac{\partial W}{\partial \alpha_{\alpha\beta}} = \frac{Et}{1-\nu^2} H^{\alpha\beta\gamma\delta}\alpha_{\alpha\beta} \tag{5.13a}$$

$$m^{\alpha\beta} = \frac{\partial W}{\partial \beta_{\alpha\beta}} = \frac{Et^3}{12(1-\nu^2)} H^{\alpha\beta\gamma\delta}\beta_{\alpha\beta} \tag{5.13b}$$

Voigt's notation can be used to reduce the order of a symmetric tensor and express it as an array. The symmetric second-order tensor

$$\boldsymbol{x} = \begin{bmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{bmatrix}$$

is expressed in Voigt's notation as $\boldsymbol{x} = [x_{11}\ x_{12}\ x_{22}]^{\mathsf{T}}$. Voigt's notation can be used to express the tensors from 5.8 and 5.13 in the simplified notation; $\boldsymbol{\alpha} = [\alpha^{11}\ \alpha^{12}\ \alpha^{22}]^{\mathsf{T}}$, $\boldsymbol{\beta} = [\beta^{11}\ \beta^{12}\ \beta^{22}]^{\mathsf{T}}$, $\boldsymbol{n} = [n^{11}\ n^{12}\ n^{22}]^{\mathsf{T}}$ and $\boldsymbol{m} = [m^{11}\ m^{12}\ m^{22}]^{\mathsf{T}}$. The expressions for membrane (5.13a) and bending stresses (5.13a) now simplify to

$$\boldsymbol{n} = \frac{Et}{1-\nu^2}\boldsymbol{H}\boldsymbol{\alpha} \tag{5.14a}$$

$$\boldsymbol{m} = \frac{Et^3}{12(1-\nu^2)}\boldsymbol{H}\boldsymbol{\beta} \tag{5.14b}$$

where $\boldsymbol{H}$ is given by;

$$\boldsymbol{H} = \begin{bmatrix} (\bar{a}^{11})^2 & \bar{a}^{11}\bar{a}^{12} & \nu\bar{a}^{11}\bar{a}^{22} + (1-\nu)(\bar{a}^{12})^2 \\ & \frac{1}{2}[(1-\nu)\bar{a}^{11}\bar{a}^{22} + (1+\nu)(\bar{a}^{12})^2] & \bar{a}^{22}\bar{a}^{12} \\ sym & & (\bar{a}^{22})^2 \end{bmatrix} \tag{5.15}$$

The weak form can be obtained by minimising the energy functional in (5.9) using

variational calculus

$$D\Pi(\boldsymbol{u})[\delta\boldsymbol{u}] = D\Pi^{int}(\boldsymbol{u})[\delta\boldsymbol{u}] + D\Pi^{ext}(\boldsymbol{u})[\delta\boldsymbol{u}] = 0 \tag{5.16}$$

The variation of internal and external energy functionals are given by;

$$\begin{aligned} D\Pi^{int}(\boldsymbol{u})[\delta\boldsymbol{u}] &= \int_\Omega n^{\alpha\beta}\delta\alpha_{\alpha\beta} + m^{\alpha\beta}\delta\beta_{\alpha\beta}d\Omega \\ &= \int_\Omega \left[\frac{Et}{1-\nu^2}\delta\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha} + \frac{Et^3}{12(1-\nu^2)}\delta\boldsymbol{\beta}^T\boldsymbol{H}\boldsymbol{\beta}\right]d\Omega \qquad (5.17a) \\ D\Pi^{ext}(\boldsymbol{u})[\delta\boldsymbol{u}] &= -\int_\Omega \boldsymbol{q}\cdot\delta\boldsymbol{u}d\Omega - \int_\Gamma \boldsymbol{N}\cdot\delta\boldsymbol{u}ds \qquad (5.17b) \end{aligned}$$

Substituting in to (5.16), the weak form can be obtained as

$$\int_\Omega \left[\frac{Et}{1-\nu^2}\delta\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha} + \frac{Et^3}{12(1-\nu^2)}\delta\boldsymbol{\beta}^T\boldsymbol{H}\boldsymbol{\beta}\right]d\Omega = \int_\Omega \boldsymbol{q}\cdot\delta\boldsymbol{u}d\Omega + \int_\Gamma \boldsymbol{N}\cdot\delta\boldsymbol{u}ds \tag{5.18}$$

5.2. Subdivision shells

First introduced in Cirak et al. [33], the shell mid-surface can be parameterised as a subdivision surface resulting in a finite element formulation with subdivision basis functions. The position and displacement fields of the shell mid-surface in a finite element are approximated with

$$\boldsymbol{x_h}(\boldsymbol{\theta_1},\boldsymbol{\theta_2}) = \sum_{i=1}^{k} N_i(\theta_1,\theta_2)\boldsymbol{x}_i \tag{5.19a}$$

$$\boldsymbol{u_h}(\boldsymbol{\theta_1},\boldsymbol{\theta_2}) = \sum_{i=1}^{k} N_i(\theta_1,\theta_2)\boldsymbol{u}_i \tag{5.19b}$$

where k denotes the number of subdivision shape functions overlapping the element and $\boldsymbol{x}_i$, $\boldsymbol{u}_i$ are the position and displacement vectors of the corresponding subdivision control points. In this context the element nodes are the control points (vertices) of the subdivision surface. The basis functions N_i and their derivatives can be evaluated as explained in Appendix C. Loop and Catmull-Clark subdivision schemes are used for finite element discretisation of triangular and quadrilateral meshes respectively. Extended subdivision rules based on Biermann et al. [14] are used to enforce special surface features

such as creases and boundaries. A detailed description of subdivision shape functions using such extended subdivision rules is given in Long [90] and Cirak and Long [31].

Expressions for the membrane and bending strains $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (5.8) can now be approximated using the above approximations

$$\boldsymbol{\alpha}_h(\theta^1,\theta^2) = \sum_{i=1}^{k} \boldsymbol{M}^i(\theta^1,\theta^2)\boldsymbol{u}_i \tag{5.20}$$

$$\boldsymbol{\beta}_h(\theta^1,\theta^2) = \sum_{i=1}^{k} \boldsymbol{B}^i(\theta^1,\theta^2)\boldsymbol{u}_i \tag{5.21}$$

The membrane-strain matrix $\boldsymbol{M}^i$ and the bending strain matrix $\boldsymbol{B}^i$ are of the form;

$$\boldsymbol{M}^i = \begin{bmatrix} N_{i,1}\boldsymbol{a}_1\cdot\boldsymbol{e}_1 & N_{i,1}\boldsymbol{a}_1\cdot\boldsymbol{e}_2 & N_{i,1}\boldsymbol{a}_1\cdot\boldsymbol{e}_3 \\ (N_{i,2}\boldsymbol{a}_1+N_{i,1}\boldsymbol{a}_2)\cdot\boldsymbol{e}_1 & (N_{i,2}\boldsymbol{a}_1+N_{i,1}\boldsymbol{a}_2)\cdot\boldsymbol{e}_2 & (N_{i,2}\boldsymbol{a}_1+N_{i,1}\boldsymbol{a}_2)\cdot\boldsymbol{e}_3 \\ N_{i,2}\boldsymbol{a}_2\cdot\boldsymbol{e}_1 & N_{i,2}\boldsymbol{a}_2\cdot\boldsymbol{e}_2 & N_{i,2}\boldsymbol{a}_2\cdot\boldsymbol{e}_3 \end{bmatrix} \tag{5.22a}$$

$$\boldsymbol{B}^i = \begin{bmatrix} \boldsymbol{B}^i_1\cdot\boldsymbol{e}_1 & \boldsymbol{B}^i_1\cdot\boldsymbol{e}_2 & \boldsymbol{B}^i_1\cdot\boldsymbol{e}_3 \\ \boldsymbol{B}^i_2\cdot\boldsymbol{e}_1 & \boldsymbol{B}^i_2\cdot\boldsymbol{e}_2 & \boldsymbol{B}^i_2\cdot\boldsymbol{e}_3 \\ \boldsymbol{B}^i_3\cdot\boldsymbol{e}_1 & \boldsymbol{B}^i_3\cdot\boldsymbol{e}_2 & \boldsymbol{B}^i_3\cdot\boldsymbol{e}_3 \end{bmatrix} \tag{5.22b}$$

where the components of the bending strain matrix are given by

$$\begin{aligned} \boldsymbol{B}^i_1 &= -N_{i,11}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[N_{i,1}\boldsymbol{a}_{1,1}\times\boldsymbol{a}_2 + N_{i,2}\boldsymbol{a}_1\times\boldsymbol{a}_{1,1} + \\ &\qquad \boldsymbol{a}_3\cdot\boldsymbol{a}_{1,1}(N_{i,1}\boldsymbol{a}_2\times\boldsymbol{a}_3 + N_{i,2}\boldsymbol{a}_3\times\boldsymbol{a}_1)] \\ \frac{\boldsymbol{B}^i_2}{2} &= -N_{i,12}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[N_{i,1}\boldsymbol{a}_{1,2}\times\boldsymbol{a}_2 + N_{i,2}\boldsymbol{a}_1\times\boldsymbol{a}_{1,2} + \\ &\qquad \boldsymbol{a}_3\cdot\boldsymbol{a}_{1,2}(N_{i,1}\boldsymbol{a}_2\times\boldsymbol{a}_3 + N_{i,2}\boldsymbol{a}_3\times\boldsymbol{a}_1)] \\ \boldsymbol{B}^i_3 &= -N_{i,22}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[N_{i,1}\boldsymbol{a}_{2,2}\times\boldsymbol{a}_2 + N_{i,2}\boldsymbol{a}_1\times\boldsymbol{a}_{2,2} + \\ &\qquad \boldsymbol{a}_3\cdot\boldsymbol{a}_{2,2}(N_{i,1}\boldsymbol{a}_2\times\boldsymbol{a}_3 + N_{i,2}\boldsymbol{a}_3\times\boldsymbol{a}_1)] \end{aligned}$$

The weak form in 5.18 can be expressed in matrix form to give the standard equilibrium equation;

$$\mathsf{K}_h\mathbf{u}_h = \mathbf{f}_h \tag{5.23}$$

The stiffness matrix and the force vector for a mesh with n_e elements are given by;

$$\mathbf{K}_h = \sum_{k=1}^{n_e} \int_{\Omega_k} \frac{Eh}{1-\nu^2} (\boldsymbol{M}^i)^T \boldsymbol{H} \boldsymbol{M}^j + \frac{Eh^3}{12(1-\nu^2)} (\boldsymbol{B}^i)^T \boldsymbol{H} \boldsymbol{B}^j d\Omega \tag{5.24a}$$

$$\mathbf{f}_h = \sum_{k=1}^{n_e} \int_{\Omega_k} \boldsymbol{q} N_i d\Omega + \int_{\Gamma_k \cap \Gamma} \boldsymbol{N} N_i ds \tag{5.24b}$$

5.3. Discrete shape derivative

Following the multiresolution approach introduced in Section 4.2, the design geometry is represented by the fine resolution finite element mesh. The shape derivatives need to be computed as a vector quantity at each node in this mesh. Based on the discretisation of the shell mid-surface using subdivision shape functions (5.19), a design change in shape can be achieved by changing the position of a subdivision control point, i.e. a node in the finite element mesh. In this context the discrete sensitivity analysis method, where the shape is first discretised to discrete design variables (Figure 4.2), can be used. Considering the format of the equilibrium condition (5.23) and the discrete nature of the optimisation variables $\mathbf{x}$, the optimisation problem (1.1) is stated as;

$$\text{minimise} \quad \mathcal{J}(\mathbf{x}, \mathbf{u}(\mathbf{x})) \tag{5.25a}$$

$$\text{subject to} \quad \mathbf{K}\mathbf{u} = \mathbf{f} \tag{5.25b}$$

where $\mathbf{u}(\mathbf{x})$ is the state variable, i.e. displacement, which depends on the design variable $\mathbf{x}$. Note that it is possible to have other equality or inequality constraints in (5.25), such as area or volume constraints. Global numerical differentiation with a finite difference scheme can be used to evaluate the design sensitivity (1.2). For example, using forward difference

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\mathbf{x}}(\mathbf{x}, \mathbf{u}(\mathbf{x})) \approx \frac{\mathcal{J}(\mathbf{x} + \Delta\mathbf{x}, \mathbf{u}(\mathbf{x} + \Delta\mathbf{x})) - \mathcal{J}(\mathbf{x}, \mathbf{u}(\mathbf{x}))}{|\Delta\mathbf{x}|} \tag{5.26}$$

The ease of implementation of this approach is offset by its computational inefficiency needing a solution cycle per each discrete design variable. Additionally the numerical errors caused by finite difference approximations can lead to severe inaccuracies. Specifically, for larger step sizes the truncation error is predominant and for smaller step sizes the roundoff error is predominant [8, 103].

Alternatively, the constrained optimisation problem (5.25) can be expressed as an equivalent unconstrained problem via a *Lagrangian function*

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = \mathcal{J}(\mathbf{u}, \mathbf{x}) + \boldsymbol{\lambda}^{\mathsf{T}}[\mathbf{f} - \mathbf{K}\mathbf{u}] \tag{5.27}$$

using the Lagrange parameter vector $\boldsymbol{\lambda}$. The necessary conditions for a minimum of the constrained problem (5.25), known as *Kuhn-Tucker conditions*, are given by the stationary points of the Lagrangian [64], i.e. $\delta\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = 0$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = 0 \quad \rightarrow \quad \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{K} = \frac{\partial \mathcal{J}}{\partial \mathbf{u}} \tag{5.28a}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = 0 \quad \rightarrow \quad \mathbf{K}\mathbf{u} = \mathbf{f} \tag{5.28b}$$

The discrete shape derivative can be expressed using the chain rule;

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{x}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{x}} + \frac{\partial \mathcal{L}}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}} \\ &= \frac{\partial \mathcal{J}}{\partial \mathbf{x}} + \left[\boldsymbol{\lambda}^{\mathsf{T}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \boldsymbol{\lambda}^{\mathsf{T}}\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \cdot \mathbf{u}\right] + \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\left[\frac{\partial \mathcal{J}}{\partial \mathbf{u}} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{K}\right] \end{aligned} \tag{5.29}$$

There exist two main methods for solving (5.29); the direct method and the adjoint method. In the adjoint method, the Lagrange multipliers $\boldsymbol{\lambda}$ can be selected such that all $\partial\mathbf{u}/\partial\mathbf{x}$ terms in (5.29) are zero, forming the adjoint problem (5.28a). The direct method involves differentiating (5.25b) with respect to $\mathbf{x}$;

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial \mathbf{x}} = \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}} - \frac{\partial \mathbf{K}}{\partial \mathbf{x}}\mathbf{u} \tag{5.30}$$

which can be solved for $\partial\mathbf{u}/\partial\mathbf{x}$. Alternatively, a similar finite difference setup to (5.26) can also be used to numerically evaluate (5.30) leading to semi-analytical sensitivity analysis.

The present work uses compliance minimisation, which is related to maximising the stiffness, as the objective function;

$$\mathcal{J} = \frac{1}{2}\mathbf{u}^{\mathsf{T}}\mathbf{K}\mathbf{u} \tag{5.31}$$

giving the following adjoint problem (5.28a);

$$\begin{aligned} \frac{\partial \frac{1}{2}\mathbf{u}^\mathsf{T}\mathbf{K}\mathbf{u}}{\partial \mathbf{u}} - \boldsymbol{\lambda}^\mathsf{T}\mathbf{K} = 0 \\ \mathbf{u}^\mathsf{T}\mathbf{K} - \boldsymbol{\lambda}^\mathsf{T}\mathbf{K} = 0 \end{aligned} \tag{5.32}$$

It is clear $\boldsymbol{\lambda} = \mathbf{u}$ solves the adjoint problem in this case and substitution in (5.29) results in the following shape derivative;

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathbf{x}} &= \frac{\partial \frac{1}{2}\mathbf{u}^\mathsf{T}\mathbf{K}\mathbf{u}}{\partial \mathbf{x}} + \left[\boldsymbol{\lambda}^\mathsf{T}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \boldsymbol{\lambda}^\mathsf{T}\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \cdot \mathbf{u}\right] \\ &= \frac{1}{2}\mathbf{u}^\mathsf{T}\frac{\partial \mathbf{K}}{\partial \mathbf{x}}\mathbf{u} + \mathbf{u}^\mathsf{T}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \mathbf{u}^\mathsf{T}\frac{\partial \mathbf{K}}{\partial \mathbf{x}}\mathbf{u} \\ &= \mathbf{u}^\mathsf{T}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \frac{1}{2}\mathbf{u}^\mathsf{T}\frac{\partial \mathbf{K}}{\partial \mathbf{x}}\mathbf{u} \end{aligned} \tag{5.33}$$

Note that there are no global operations in the above expression and the partial derivatives of the stiffness matrix and the force vector for Kirchhoff-Love shells (5.24) can be assembled element-wise for each design variable $\boldsymbol{x}_i$;

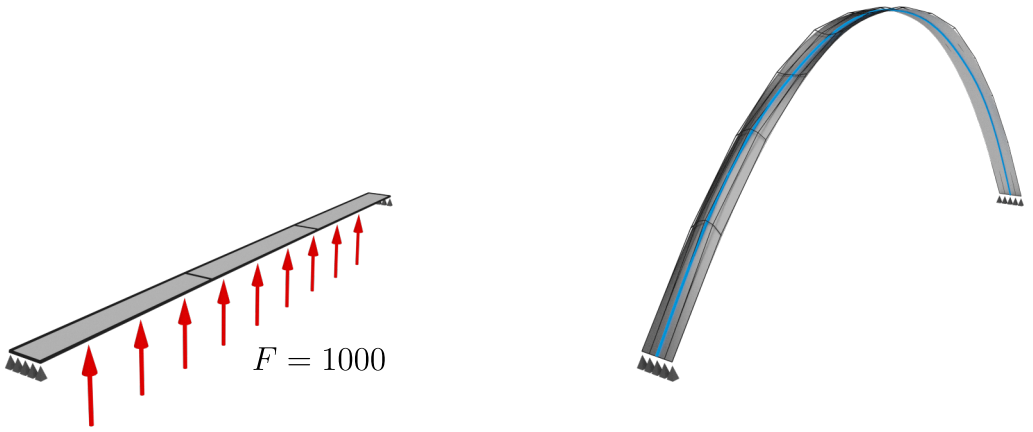
$$\begin{aligned} \frac{\partial \boldsymbol{K}_h}{\partial \boldsymbol{x}_i} = \sum_{k=1}^{n_e} \int_{\Omega_k} & \frac{Eh}{1-\nu^2}\left[\frac{\partial (\boldsymbol{M}^i)^\mathsf{T}}{\partial \boldsymbol{x}_i}\boldsymbol{H}\boldsymbol{M}^j + (\boldsymbol{M}^i)^\mathsf{T}\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{x}_i}\boldsymbol{M}^j + (\boldsymbol{M}^i)^\mathsf{T}\boldsymbol{H}\frac{\partial \boldsymbol{M}^j}{\partial \boldsymbol{x}_i}\right] \\ & + \frac{Eh^3}{12(1-\nu^2)}\left[\frac{\partial (\boldsymbol{B}^i)^\mathsf{T}}{\partial \mathbf{x}_i}\boldsymbol{H}\boldsymbol{B}^j d\Omega + (\boldsymbol{B}^i)^\mathsf{T}\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{x}_i}\boldsymbol{B}^j + (\boldsymbol{B}^i)^\mathsf{T}\boldsymbol{H}\frac{\partial \boldsymbol{B}^j}{\partial \boldsymbol{x}_i}\right] d\Omega \\ & + \left[\frac{Eh}{1-\nu^2}(\boldsymbol{M}^i)^\mathsf{T}\boldsymbol{H}\boldsymbol{M}^j + \frac{Eh^3}{12(1-\nu^2)}(\boldsymbol{B}^i)^\mathsf{T}\boldsymbol{H}\boldsymbol{B}^j\right]\frac{\partial d\Omega}{\partial \boldsymbol{x}_i} \end{aligned} \tag{5.34}$$

$$\frac{\partial \boldsymbol{f}_h}{\partial \boldsymbol{x}_i} = \sum_{k=1}^{n_e} \int_{\Omega_k} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}_i} N_i d\Omega + \boldsymbol{q} N_i \frac{\partial d\Omega}{\partial \boldsymbol{x}_i} + \int_{\Gamma_k \cap \Gamma} \boldsymbol{N} N_i \frac{\partial ds}{\partial \boldsymbol{x}_i} \tag{5.35}$$

Note that the above require the partial shape derivatives of the covariant and contravariant basis vectors which is explained in Appendix D.1.

5.4. Examples

Three examples are presented to demonstrate the functioning of the proposed multiresolution frameworks for shape optimisation of Kirchhoff-Love shells. The objective is to minimise the structural compliance (5.31). The discretised optimisation problem is



(a) Initial geometry. All coarse resolution nodes in $\mathbf{x}^{\ell_c}$ along the supported edges are pinned as indicated. The mesh indicates the optimisation level ℓ_o which is 0 initially.

(b) Optimised geometry. The blue line shows the centre line of the optimised strip used in the catenary curve comparison in figure 5.3.The mesh indicates the optimisation resolution, now at $\ell_o = 2$.

Figure 5.2.: Optimisation of thin strip to form a catenary curve. Initial and optimised strip for a curve with $l = 1.3$ and supports at equal height.

solved with the Method of Moving Asymptotes (MMA) [124] as implemented in the nlopt library [76]. The input to the nlopt library consists of the cost function $\mathcal{J}(\mathbf{x}^{\ell_c}, \mathbf{u}(\mathbf{x}^{\ell_c}))$ and the position vectors $\boldsymbol{x}_i^{\ell_o}$ and the gradient $\boldsymbol{g}_i^{\ell_o}$ for each coarse geometry node. In the subdivision framework, the coarse resolution gradients are obtained by coarsening the fine resolution gradients $\boldsymbol{g}_i^{\ell_c} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_i^{\ell_c}}$ as described in Section 4.3.3. In addition, for some examples, geometric bounds for the design variables and additional area constraints are provided to the nlopt library. During optimisation the position vectors are updated using the relevant multiresolution algorithm 1 or 2.

5.4.1. Catenary curve

In this verification example, a thin strip pinned at the ends is optimised for a vertical pressure load and compared with a catenary curve. It has been known since the work of Leibniz, Huygens and Johann Bernoulli in 1691 that the shape of a curve assumed by a loose string hung freely from two fixed points is a catenary [89]. The equation of the

catenary curve reads

$$y = \alpha \cosh\left(\frac{x+\beta}{\alpha}\right) + \gamma \tag{5.36}$$

The three constants α, β and γ are determined based on the location of the supports and length of the string l. Two cases are studied, one where the supports are at equal height and the other where one support is higher than the other. In both cases the horizontal distance between the supports is 1. The vertical offset between the supports in the second case is 0.2. Initially, the strip is a flat plate connecting the supports with length equal to the distance between the supports. Width and thickness are constant at 0.05 and 0.02 respectively. The magnitude of vertical uniformly distributed load is 1000, the Young's modulus and Poisson's ratio are respectively 2×10^8 and 0.3. Length of the curve l is varied at 1.1, 1.2 and 1.3. This is realised by using an area constraint during optimisation;

$$0.05l - A_\tau = 0 \tag{5.37}$$

where $A_\tau = \int J$ denotes the current area of the strip with Jacobian of the mapping $J = |\boldsymbol{a}_1 \times \boldsymbol{a}_2|$. Note that the derivative of the area constraint (6.16) is now required at each design variable. The derivative of J is computed (see Appendix D.1) for each fine resolution node $\boldsymbol{x}_i$ and projected to the coarse resolution in the same way as the shape derivative. Notice that this constraint is initially violated in all cases.

Multiresolution geometry description: Catmull-Clark subdivision

Extended Catmull-Clark subdivision is used for creating the multiresolution framework and solving the mechanical problem via subdivision shells. The initial coarse mesh used for optimisation contains only 3 elements as shown in Figure 5.2a. This increases to 48 in the twice subdivided fine mesh used for computations. During multiresolution optimisation, the optimisation resolution is increased from $\ell_o = 0$ until $\ell_o = 2$ is reached. Note that corner nodes of the strip have been tagged as corner to maintain the rectangular shape during subdivision refinement and computation of subdivision shape functions. Only the y coordinates are optimised and as a result, only in-plane shape changes are expected. Comparison of the optimisation results with the corresponding catenary curve for different curve lengths and support conditions is shown in Figure 5.3. The reduction of the objective function is more than 99.9% for all cases and the results show good visual agreement with catenary curves. It is important that transverse bending (bending about the centre line in Figure 5.2a) in the strip is minimised to be faithful to the beam assumption, hence the relatively large thickness value. However it is seen from

figure 5.2 that optimisation has resulted in some lateral curvature as it would stiffen the strip. This indicates a small deviation from beam behaviour and explains why a perfect catenary profile is not obtained. Ideally, the optimisation problem needs to be solved using a beam formulation for a better comparison with a caternary profile.

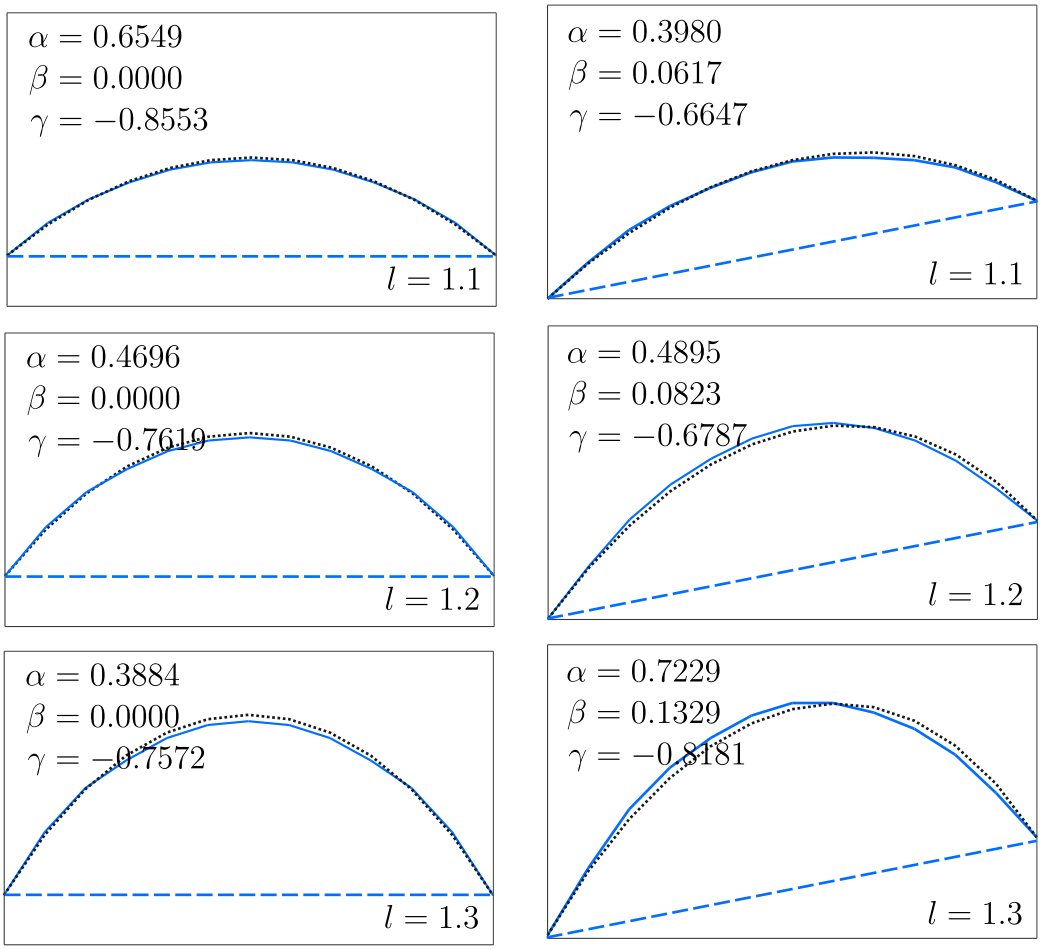
5.4.2. Bi-parabolic roof shell

This example is adapted from Bletzinger and Ramm [16] where parameter optimisation was used to optimise a roof with rectangular plan (6×12). The roof is of thickness 0.05 with pinned supports along the short edges and loaded by a vertical load of 5000. The Young's modulus and the Poisson's ratio are respectively 3×10^{10} and 0.2. In the reference problem two parameters s_1, s_2 (Figure 5.4) are used to change the roof shape while maintaining a bi-parabolic shape.

Multiresolution geometry descriptions: Catmull-Clark and Loop subdivision Unlike the reference problem where the initial geometry is a cylindrical shell (Figure 5.4a), a flat rectangle is used with several different mesh layouts as the initial geometry to highlight mesh dependence of the optimised shape. During multiresolution shape optimisation, ℓ_o is initiated at 0 and gradually increased to 2, equal to the computational level ℓ_c. Only the out-of-plane position of the vertices are optimised with an upper bound of 6 to reproduce the effect of limiting the maximum value of s_1 to 6 in the reference problem.

The different meshes used and the resulting optimised shapes are shown in Figure 5.5. The corner nodes in each mesh have been tagged as corner to maintain the rectangular shape during subdivision refinement and computation of subdivision shape functions. For comparison of each optimal shape with the reference, the objective function is evaluated for a very fine resolution mesh at the end of optimisation. This evaluation of cost from a converged solution for each shape is necessary since each starting mesh is different with respect to element size and mesh layout. In this setting, the cost of the initial and optimised shape of the reference solution is 2108 and 149.68 respectively while the flat plate used here as the initial shape has a cost $\approx 46 \times 10^4$.

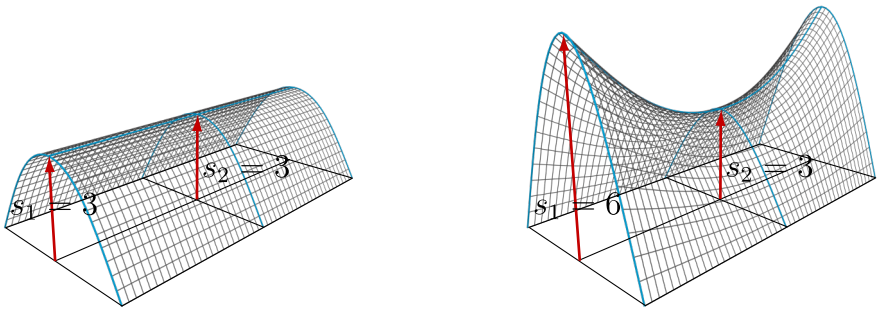
It is clear from the results that there are several local minima in this problem resulting in different optimised shapes depending on the initial mesh used. Only having the minimal number of design variables at the initial resolution $\ell_o = 0$, as in mesh A and



(a) Supports at same height

(b) Supports at different heights

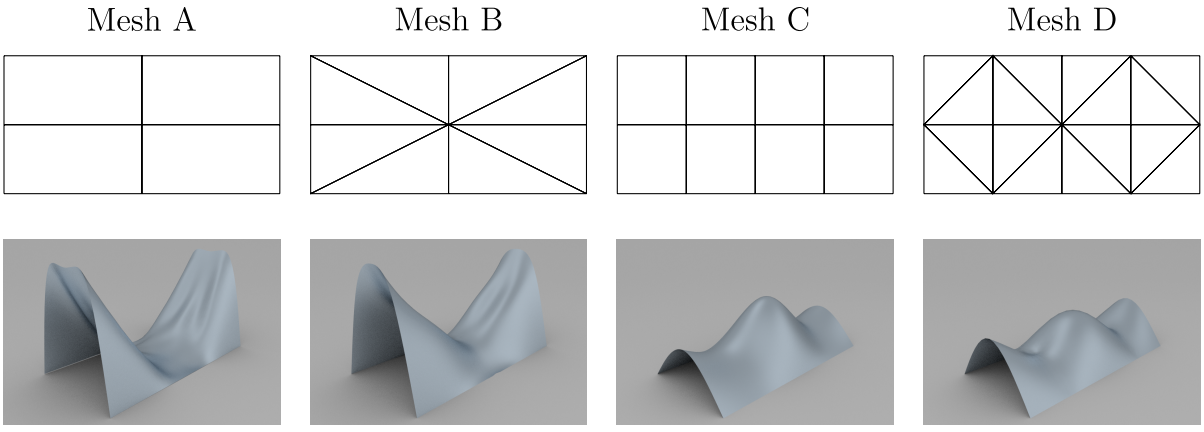
Figure 5.3.: Optimised catenary curve shapes. The blue lines show the centre lines of the strip before and after optimisation with the solid blue line indicating the latter. The dotted black line is the catenary curve of corresponding length l for each case. The α, β and γ parameters in (5.36) for given curve length and support positions are as indicated. The origin is such that the left support has coordinates $(-0.5, 0.0)$ in all cases.



(a) Initial geometry
Cost = 2108

(b) Optimised geometry
Cost = 149.68

Figure 5.4.: Bi-parabolic roof shell, initial and optimised roof from Bletzinger and Ramm [16]. The shape is defined using four parabolic curves indicated as blue lines. The parabolic curves are parameterised by the design variables s_1 and s_2. The final value of the objective function (cost) for each optimised shape, recomputed from the subdivision shell solver for comparison, is as indicated.



cost = 39.04 cost = 54.36 cost = 338.48 cost = 376.24

Figure 5.5.: Roof shell optimised using multiresolution subdivision framework with different starting meshes. The top row shows the different meshes for the starting geometry at initial resolution $\ell_o = 0$. The limit surfaces of the optimised shapes and the corresponding cost for each mesh after multiresolution optimisation are shown in the bottom row.

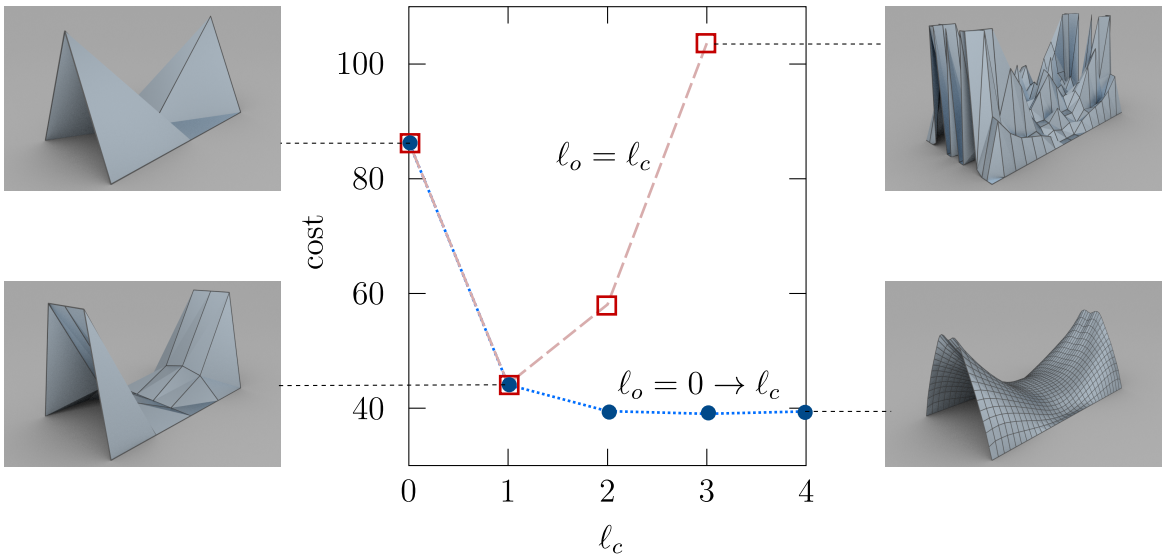
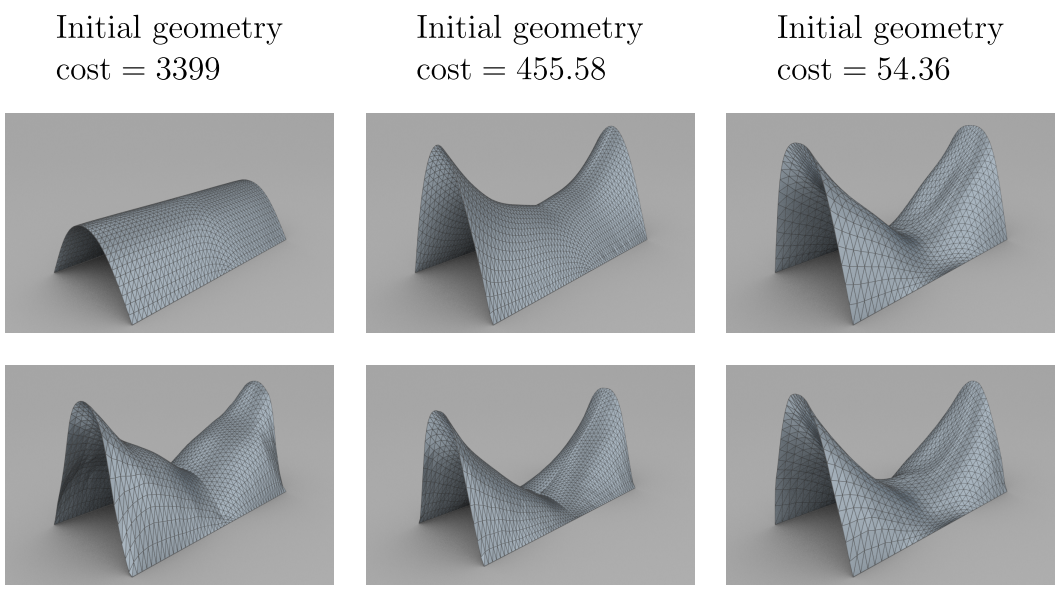


Figure 5.6.: Comparison of roof shell optimisation with multiresoluiton subdivision. The optimum shapes for mesh A using single and multiresolution are compared. The graph plots minimal cost obtained from using different computational levels during optimisation. The dashed red line indicates single resolution optimisation and the dotted blue line is for multiresolution optimisation. Each data point denotes the final cost of an independent optimisation problem with initial geometry given by mesh A and optimised with either $\ell_o = \ell_c$ or $\ell_o = 0 \to \ell_o$. Inset pictures are the optimised shapes at the computational level.

B, ensures convergence to the correct global minima which is similar to the optimised shape of the reference solution (Figure 5.4b). A comparison of single and multiresolution optimisation for mesh A is shown in Figure 5.6. In single resolution optimisation, the optimisation level is maintained at the same level as the computational level, $\ell_o = \ell_c$. Note that the final cost is computed using a much finer mesh of each optimum shape as before. A feature of this problem is that the main solution features can be closely approximated using a once subdivided mesh A, explaining the relatively good results obtained from both single and multiresolution optimisation at $\ell_c = 1$. When using finer computational levels, single resolution optimisation quickly leads to noisy solutions while multiresolution optimisation continues to provide improved solutions.

Multiresolution geometry description: progressive mesh

Optimisation with progressive meshes is not suitable for making large shape changes, such as in the previous subdivision examples where the starting geometry is a flat plate. The extreme coarse resolutions required in such cases imply large geometric details



(a) Initial fine mesh contains 1617 nodes and is geometrically similar to that in Figure 5.4a. Optimisation resolutions $\ell_o \in \{48, 80, 161\}$.

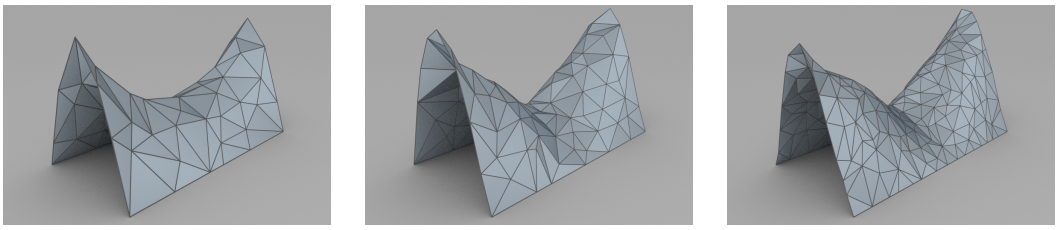
(b) Initial fine mesh contains 2145 nodes and is geometrically similar to that in Figure 5.4b. Optimisation resolutions $\ell_o \in \{64, 107, 214\}$.

(c) Initial fine mesh contains 1089 nodes and is the optimised mesh B from Figure 5.5. Optimisation resolutions $\ell_o \in \{108, 217, 326\}$.

Figure 5.7.: Bi-parabolic roof shell optimisation with progressive meshes from different initial geometries.

(Equations 3.13 and 3.14) causing undesirable effects during detail restoration. Additionally, the error due to sub-sampling of the shape derivative (Section 4.4.3) is largest when optimising design variables in such extreme coarse resolutions.

In this context, the previous roof example is optimised with progressive meshes starting from better initial geometries than a flat plate. Figure 5.7 contains several such example where the initial geometry is a given fine resolution mesh. Computations are done using extended Loop subdivision shell shape functions computed in the given fine resolution mesh. This requires the corners nodes of the fine resolution mesh being tagged as before. A progressive mesh hierarchy is built to provide the coarse resolutions nodes used as the design variables. For example, the coarse resolution $\mathbf{x}^{64}$ shown in Figure 5.8a is used as the first set of design variables in optimising the initial mesh $\mathbf{x}^{2145}$ shown in Figure 5.7b. A binary oracle is used to preserve the corner nodes during decimation and the boundaries are preserved as explained in Section 3.2.2.



(a) $\mathbf{x}^{64}$ (b) $\mathbf{x}^{107}$ (c) $\mathbf{x}^{214}$

Figure 5.8.: Multiple progressive mesh coarse resolutions used during optimisation of model shown in Figure 5.7b.

Once the shape has been optimised using the design variables in $\mathbf{x}^{64}$, the optimisation resolution is increased to $\ell_o = 107$ and the corresponding coarse resolution $\mathbf{x}^{107}$ is shown in Figure 5.8b. The process is repeated using resolution $\mathbf{x}^{214}$, shown in Figure 5.8c, after which using any finer resolutions causes wiggly and irregular optimised shapes due to local minima. Results show good agreement with the subdivision framework, i.e. optimised shapes in Figures 5.7b and 5.7c closely resemble the optimum shapes from multiresolution subdivision in Figure 5.5. However it is clear that optimising an existing shape using progressive mesh geometry produces inferior results in comparison to gradually building an optimised shape using subdivision.

5.4.3. Architectural roof design

Architects frequently require shell design where structural efficiency is compromised for aesthetics. In this example, multiresolution shape optimisation is used for design space exploration where the architect can actively seek optimal structural efficiency for given aesthetic constraints. An initial model shown in Figure 5.9 is given with approximate dimensions of $2.31 \times 6.27 \times 0.75$. A vertical uniformly distributed load of -1000 is considered as the design load and the shell thickness is set to $t = 0.02$. The Young's modulus and the Poisson's ratio are 1×10^{10} and 0.2 respectively. Only the z coordinates of nodes are used as design variables to preserve the plan shape during optimisation. An area constraint is present to restrict the size of the optimised shape;

$$A_t - 1.2A_o \leq 0 \tag{5.38}$$

where A_o and A_t denote the initial and current roof surface area. An important architectural feature of the problem is the roof ridge profile. Three different design scenarios

are considered where the profile of the central ridge is preserved to varying degree as indicated in Figure 5.10.

Multiresolution gometry description: progressive meshes
The different design constraints in Figure 5.10 pose additional requirements on the decimation and smoothing process.

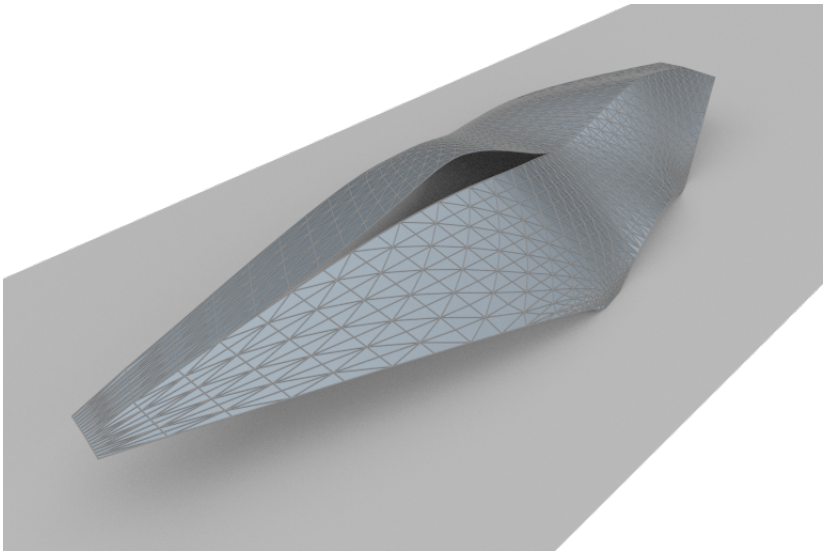
- The start and end nodes of each line support segment need to be preserved during decimation. This is easily accomplished via a binary oracle (Figure 5.11). The roof ridge is tagged as a creased edge to use a special edge preserving quadric as explained in Section 3.2.2.

- The progressive mesh shape functions are not suited for implementing Dirichlet conditions by design. Essentially the one-ring Laplacian smoothing causes propagation of geometry modification to constrained regions. However this is easily fixed by restricting the one-ring smoothing to include only neighbouring nodes belonging to the same Dirichlet support.

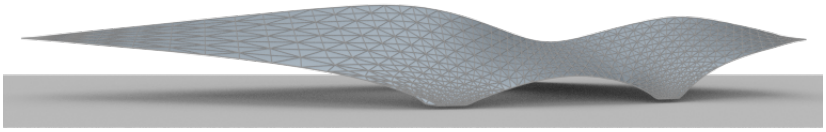
- In each design scenario a specific region of the geometry is prevented from being modified by the optimisation process. This is realised by not optimising any coarse resolution node in such regions. Additionally, the method described above for enforcing Dirichlet conditions is used to prevent propagation of geometry modifications from neighbouring nodes.

The tagged nodes and edges in Figure 5.11 are also used in construction of the extended Loop subdivision shape functions for solving the mechanical problem. Essentially the nodes marked in black used by the binary oracle are designated as corner nodes and the roof ridge as a creased edge in a subdivision setting. The fine resolution mesh $\mathbf{x}^{1797}$ used in the analysis module is decimated to obtain different optimisation resolutions $\ell_o \in \{26, 35, 89\}$ as shown in Figure 5.12.

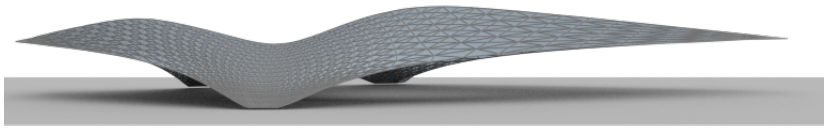
The initial value of the objective function is 24.11, the largest reduction of which is obtained in design **C** where the final cost is 63.32% lower than the original cost. Design **A** also achieves a significant cost reduction of 51.02%. In comparison, design **B** only manages a cost reduction of 28.6% due to the restricted design space. The corresponding optimised shapes for each design scenario are shown in Figures 5.13, 5.14 and 5.15.



(a) 3D view

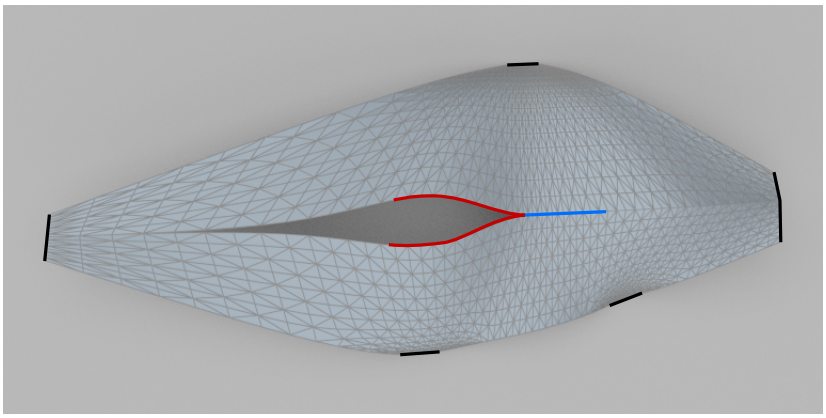


(b) Front view

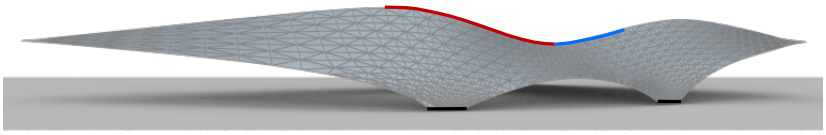


(c) Back view

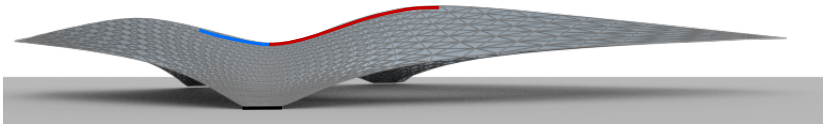
Figure 5.9.: Given initial mesh of architectural roof containing 1797 nodes. The model is asymmetric with three line supports and a central opening. A roof ridge is also present as indicated by the creased geometric edge in Figure 5.9a.



(a) Top view

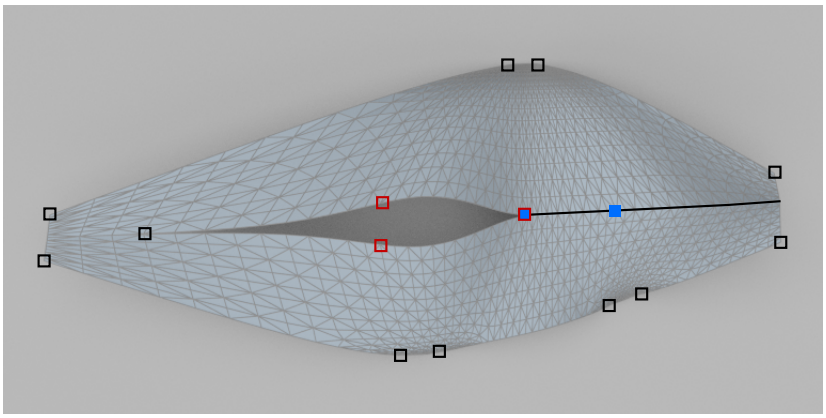


(b) Front view

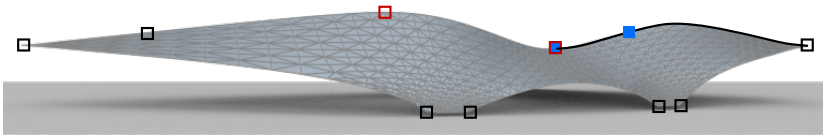


(c) Back view

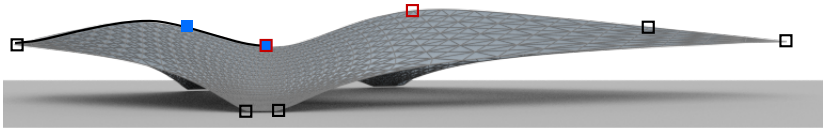
Figure 5.10.: Design constraints for the architectural roof optimisation problem. The edges marked in black are preserved in all designs in order to preserve the supports and maintain overall length. In design **A**, the edges marked in red need to be preserved while both edges in red and blue are preserved in design **B**. Only the essential black edges are preserved in design **C**.



(a) Top view

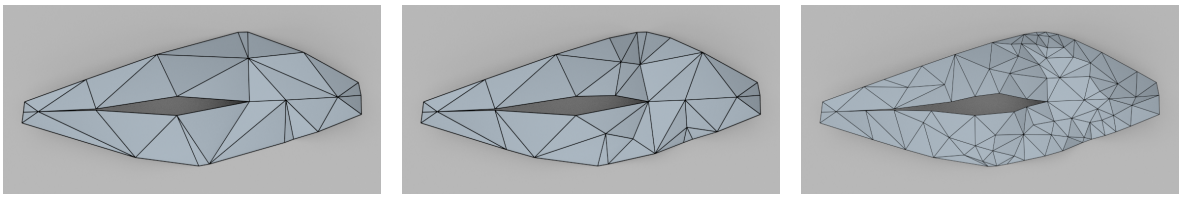


(b) Front view



(c) Back view

Figure 5.11.: Architectural roof design, special decimation conditions to preserve supports and roof ridge. A binary oracle is used to prevent the nodes indicated in black from being decimated, this is required for maintaining sharp edges and corners. Additionally, the nodes marked in colour are preserved in each of the design scenarios described in Figure 5.10. The marked edges are tagged as creased for preserving the roof ridge. Note that additional edges are marked as creased according to the design scenario indicated in Figure 5.10.



(a) $\mathbf{x}^{26}$ (b) $\mathbf{x}^{35}$ (c) $\mathbf{x}^{89}$

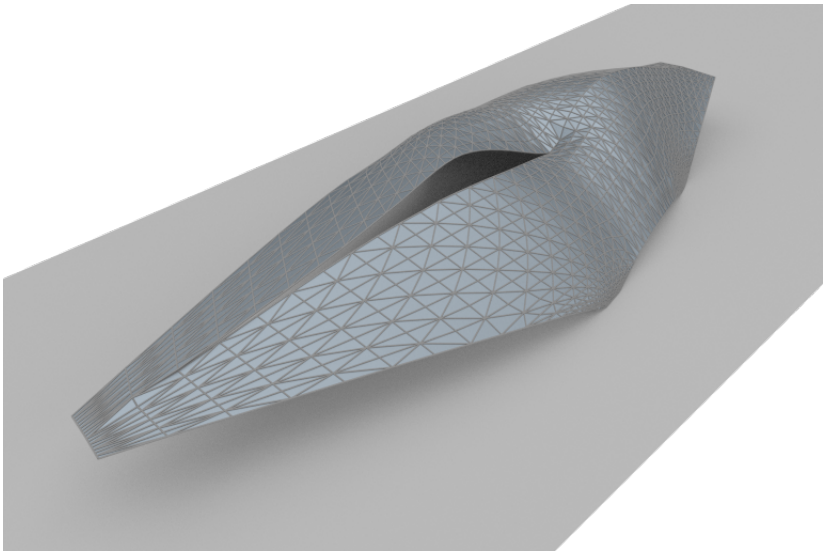
Figure 5.12.: Multiple progressive mesh coarse resolutions $\ell_o \in \{26, 35, 89\}$ used during optimisation of architectural roof model. The coarse resolutions have been created using special decimation conditions indicated in Figure 5.11. Note that these coarse resolutions have been successively generated from the initial fine resolution without involving any optimisation.

Multiresolution geometry description: Loop subdivision

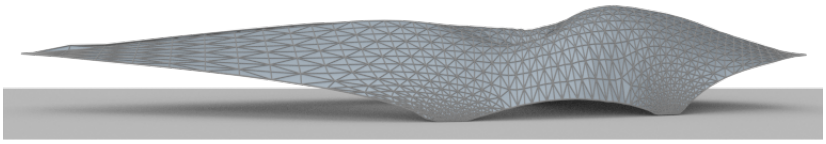
In order to use the subdivision framework for optimising the given geometry, a subdivision control mesh is manually created such that the limit surface closely resembles the original geometry. The obtained subdivision limit surface and the corresponding control mesh are shown in Figures 5.16 and 5.17. The line supports in the original model have been simplified to point supports in order to avoid coarse resolution elements with bad aspect ratios.

The design scenarios defined on the original geometry (Figure 5.10) are now expressed in an equivalent manner by tagging the coarse resolution nodes. In Figure 5.17, the nodes and vertices indicated in black are used to enforce corners and creased edges during multiresolution refinement and evaluation of extended Loop subdivision shape functions. Different design scenarios are implemented by fixing different nodes on the control mesh to exclude them from optimisation. When the coarse resolution is refined in later optimisation stages, all descendants nodes (one-ring neighbours for Loop subdivision) of such fixed nodes become fixed.

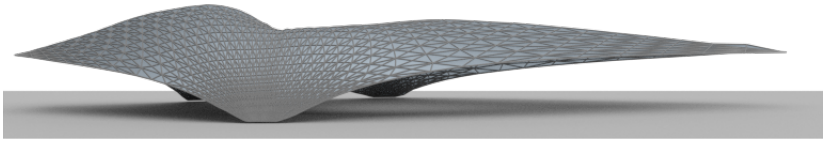
The coarse mesh at optimisation level $\ell_o = 0$ contains 26 nodes which is twice subdivided to obtain the computation mesh at level $\ell_c = 2$ with 272 nodes. The second optimisation stage is done at level $\ell_o = 1$ but no optimisation is done at level $\ell_o = 2$ as this results in local wrinkling. The initial value of the objective function is 31.36 (c.f. the starting cost in the progressive mesh model is 24.11). As in the progressive mesh case, design **C** results in the most efficient optimised shape with a 79.13% reduction is cost followed by design **A** with a reduction of 38.88%. The total reduction in the most constrained design **B** is only 23.97%. The corresponding optimised shapes for each design scenario



(a) 3D view

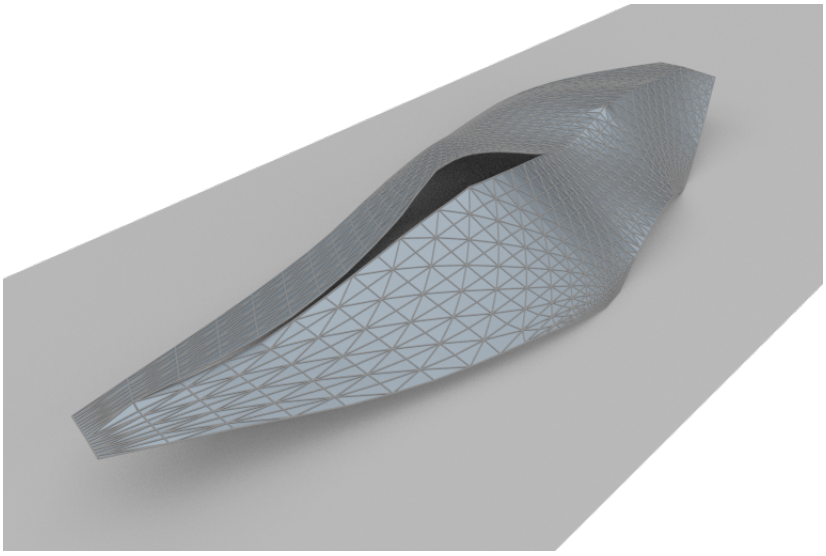


(b) Front view

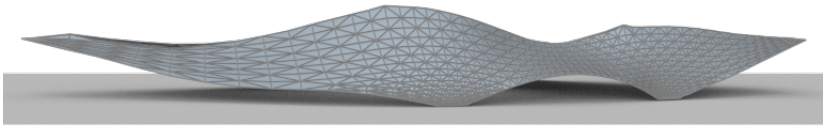


(c) Back view

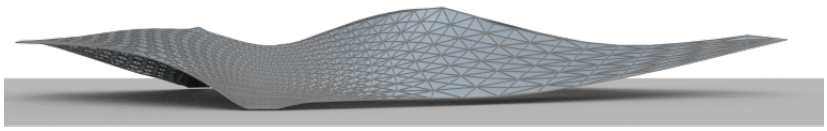
Figure 5.13.: Optimised architectural roof for design scenario **A** using progressive meshes. The final value of the objective function is 11.81, representing a 51.02% reduction.



(a) 3D view

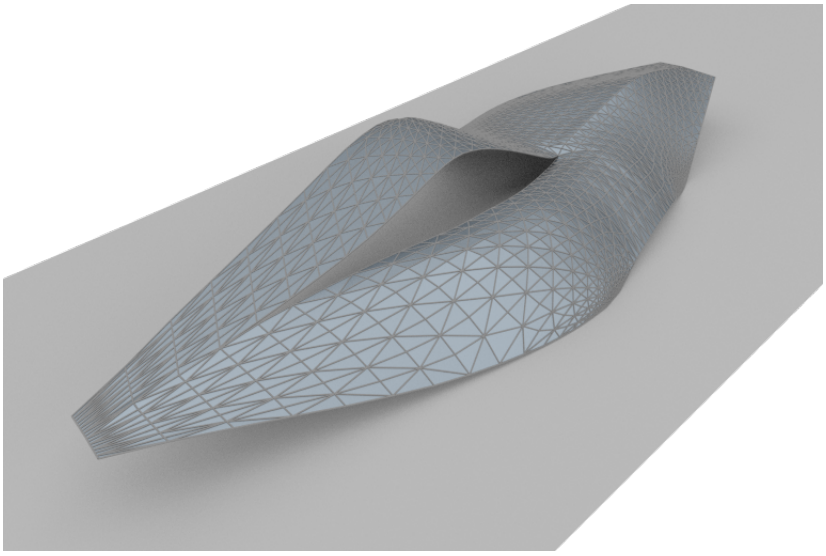


(b) Front view

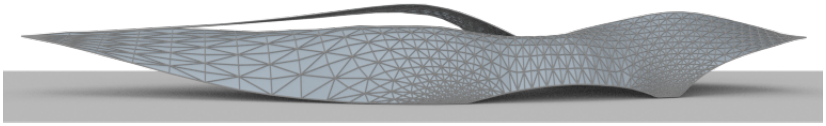


(c) Back view

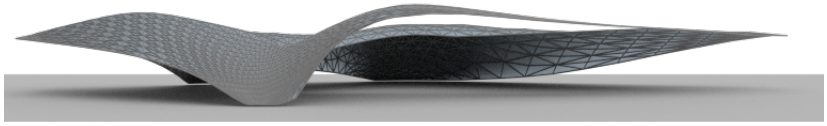
Figure 5.14.: Optimised architectural roof for design scenario **B** using progressive meshes. The final value of the objective function is 17.22, representing a 28.6% reduction.



(a) 3D view

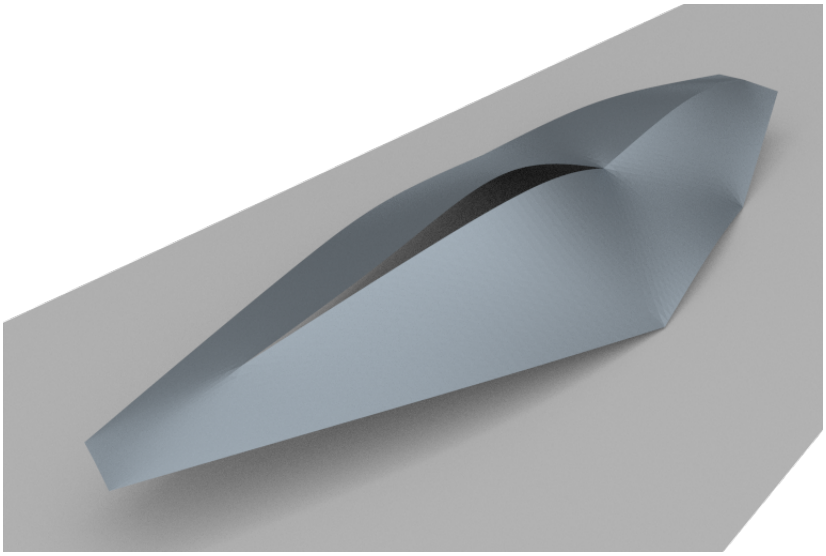


(b) Front view

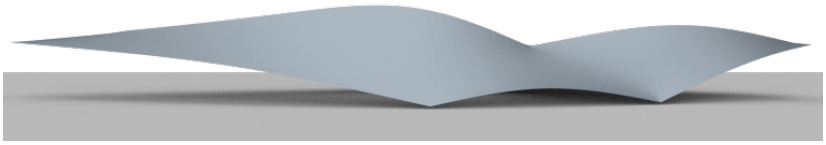


(c) Back view

Figure 5.15.: Optimised architectural roof for design scenario **C** using progressive meshes. The final value of the objective function is 8.87, representing a 63.32% reduction.



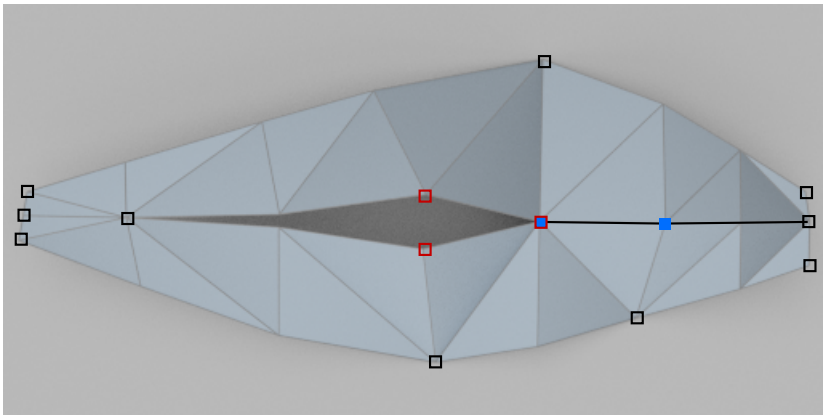
(a) 3D view



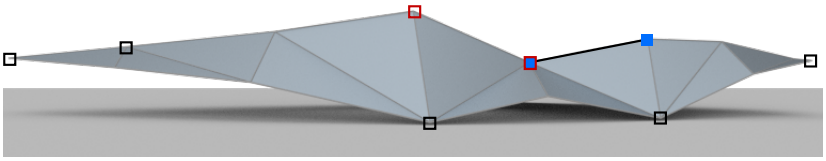
(b) Front view

(c) Back view

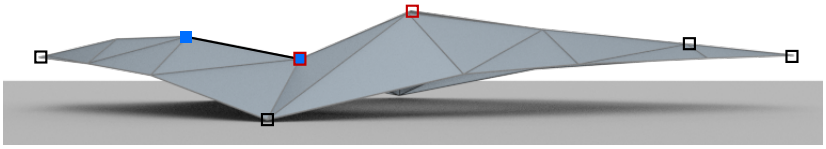
Figure 5.16.: Limit surface of the approximate subdivision surface created to resemble the original model in Figure 5.9. The limit surface is generated by applying extended Loop subdivision scheme on the coarse resolution control mesh shown in Figure 5.17.



(a) Top view



(b) Front view

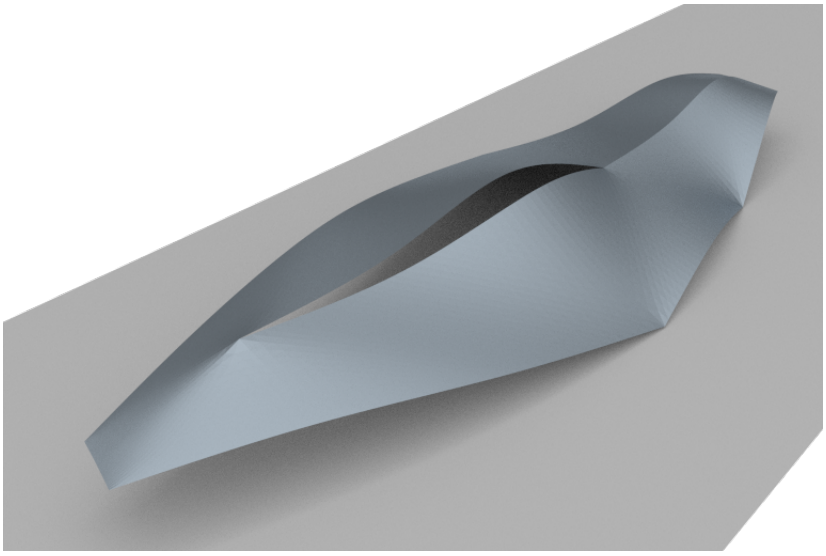


(c) Back view

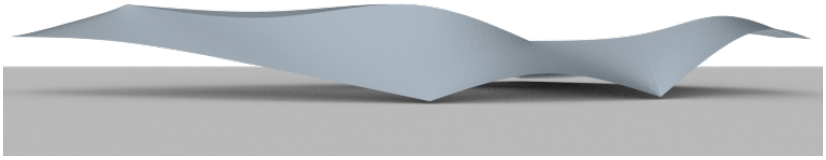
Figure 5.17.: Mesh tagging in the subdivision architectural roof model. The nodes and edges indicated in black are tagged as corner and creased respectively. This preserves sharp edges and corners as can be observed in the rendering of the limit surface shown in Figure 5.16. The coloured nodes are used to enforce the design scenarios described in Figure 5.10. The red nodes are fixed in design **A**, while both red and blue nodes are fixed in design **B**. In design **C**, only the black nodes are fixed.

are shown in Figures 5.18, 5.19 and 5.20.

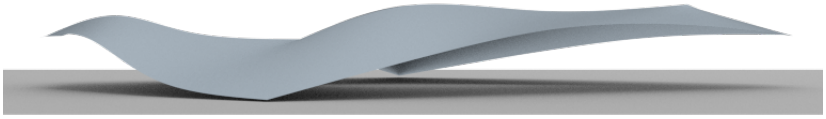
The importance of accurate representation of the original design is highlighted in the difference between the progressive mesh and subdivision results for the different design cases. However in both frameworks, the order of cost reduction follows the richness of the design space as expected. This concludes the chapter on shape optimisation of shells using subdivision and progressive mesh multiresolution methods. Shape optimisation of solids will be explored in the next chapter.



(a) 3D view

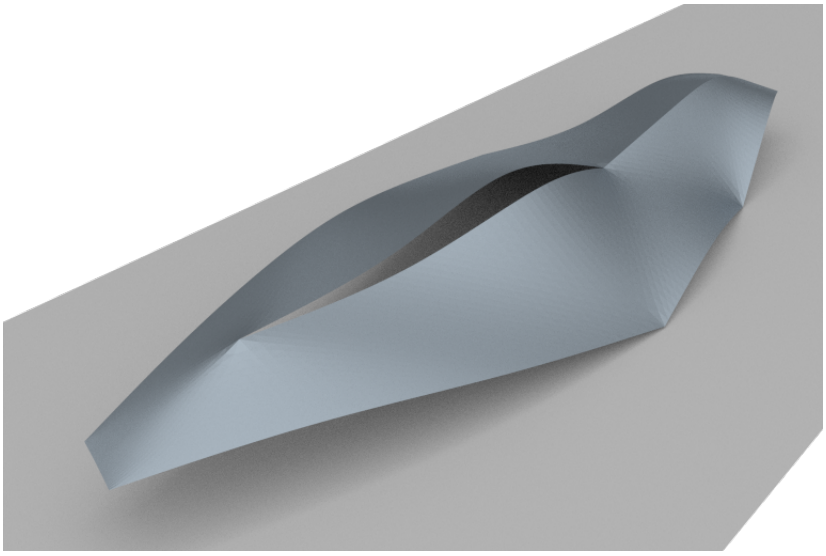


(b) Front view

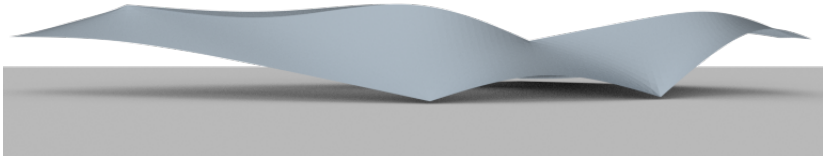


(c) Back view

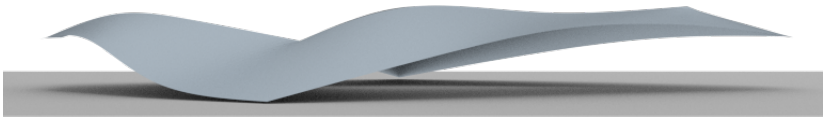
Figure 5.18.: Limit surface of the optimised architectural roof for design scenario **A** using subdivision. The final value of the objective function is 19.17 representing a 38.88% reduction.



(a) 3D view

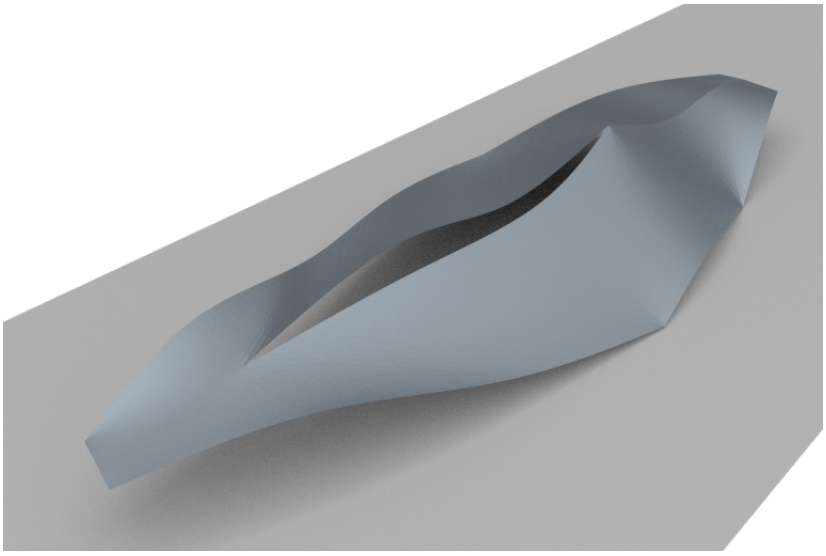


(b) Front view

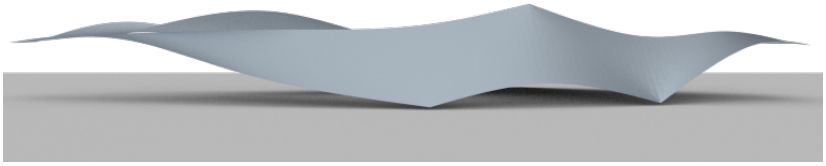


(c) Back view

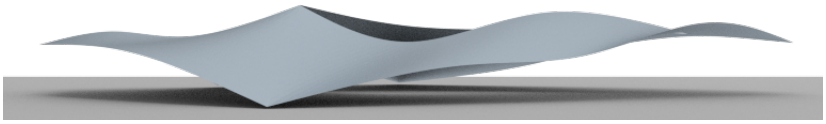
Figure 5.19.: Limit surface of the optimised architectural roof for design scenario **B** using subdivision. The final value of the objective function is 23.84 representing a 23.97% reduction.



(a) 3D view



(b) Front view



(c) Back view

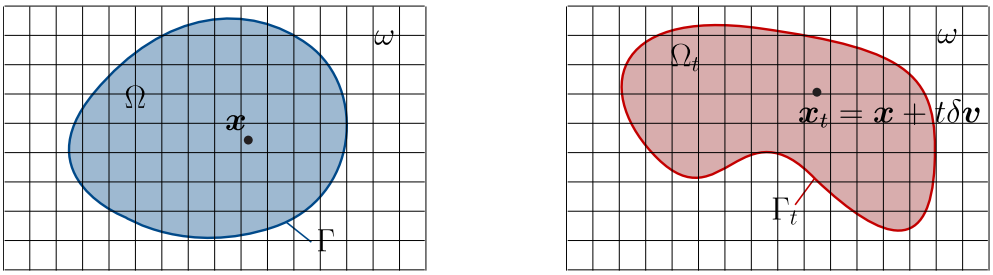
Figure 5.20.: Limit surface of the optimised architectural roof for design scenario **C** using subdivision. The final value of the objective function is 6.55 representing a 79.13% reduction.

6. Optimisation of solids

In CAD, the geometry of a solid is typically expressed using boundary representations (B-reps) or constructive solid geometry (CSG). The former expresses the volumetric object as a collection of surfaces whereas the latter uses boolean expressions on primitive solid objects [68]. The conventional finite element analysis of such solids requires a volume mesh which needs to be generated via some meshing tool while taking the B-rep or CSG model as input. During shape optimisation, any significant changes to the surface geometry leads to severe distortion of the solid mesh rapidly leading to inaccurate analysis results. This makes it necessary to frequently regenerate or to smooth the volume mesh during shape optimisation. One possible solution to this problem comes from immersed and boundary element methods that only require a surface mesh representing the domain boundary to be deformed.

The immersed finite element approach, also known as the fixed grid finite element method, involves the solution of the original problem on a proxy domain created by immersing a boundary mesh in a slightly larger Cartesian background grid (Figure 6.1). During shape optimisation, the proxy domain is updated by re-evaluating the cells cut by the new boundary and involves no remeshing. This enables the problem to be discretised on a surface mesh directly obtained from the B-rep CAD model. Additionally, the generation of surface meshes is usually substantially easier than the generation of volume meshes.

The solid optimisation framework presented in this chapter uses B-rep CAD models to represent the boundary of the solid domain. The input B-rep is either a subdivision surface mesh (Section 4.3) or a progressive mesh representation (Section 4.4). Linear elastic optimisation of solids using an immersed method is presented in Section 6.1 which is next extended to topology optimisation in Section 6.3. The boundary element method, by design, restricts the problem formulation to the boundary and offers another means of solid optimisation without necessarily using a body fitted solid mesh. A boundary



(a) Initial domain (b) Deformed domain

Figure 6.1.: Immersed finite element method for shape optimisation. The original problem domain Ω is deformed to Ω_t during shape optimisation corresponding to the boundary change $\Gamma \to \Gamma_t$. In immersed methods, a proxy problem domain is created by immersing the domain boundary Γ in a fixed background grid ω.

element method is used to demonstrate shape optimisation for electrostatic problems in Section 6.5.

6.1. Linear elastic shape optimisation

Consider the linear elastic boundary value problem for a solid body with the domain Ω

$$-\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{f} \quad \text{in } \Omega \tag{6.1a}$$

$$\boldsymbol{u} = \boldsymbol{0} \quad \text{on } \Gamma_D \tag{6.1b}$$

$$\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N \tag{6.1c}$$

where $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{f}$ is the external load vector and $\bar{\boldsymbol{t}}$ is the prescribed traction on the Neumann boundary Γ_N with the outward normal $\boldsymbol{n}$. For simplicity only homogeneous Dirichlet boundary conditions are assumed on Γ_D. The linear strain tensor $\boldsymbol{\epsilon}(\boldsymbol{u})$ is defined as follows

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla^{\mathsf{T}} \boldsymbol{u}) \tag{6.2}$$

The linear elastic material model is given by

$$\boldsymbol{\sigma}(\boldsymbol{\epsilon}) = \boldsymbol{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) \tag{6.3}$$

where $\boldsymbol{C}$ is a fourth order constitutive tensor. In the present work, compliance minimisation is used which can be expressed as

$$\mathcal{J}(\Omega, \boldsymbol{u}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{u} \, \mathrm{d}\Omega + \int_{\Gamma_N} \bar{\boldsymbol{t}} \cdot \boldsymbol{u} \, \mathrm{d}\Gamma = \frac{1}{2} \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u}) \, \mathrm{d}\Omega \tag{6.4}$$

6.1.1. Shape derivative

The present work uses shape sensitivity analysis for obtaining the shape derivative following the work of Allaire et al. [4]. A brief outline of this approach is given below, see [4, 6, 22] for more details.

Following the notation introduced in Section 4.2, let $\delta\boldsymbol{v}$ represent the velocity vector describing the deformation from the initial configuration Ω_0 to the updated configuration Ω_t. Evaluating the shape derivative (4.4) requires the directional derivative of the state variable $\boldsymbol{u}$ in the direction of the velocity vector $\delta\boldsymbol{v}$. This is clearly seen by using the chain rule

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\Omega}\delta\boldsymbol{v} = \frac{\partial \mathcal{J}}{\partial \Omega}\delta\boldsymbol{v} + \frac{\partial \mathcal{J}}{\partial u}\frac{\partial u}{\partial \Omega}\delta\boldsymbol{v} \tag{6.5}$$

Alternatively, an adjoint method can be used, as demonstrated for the discrete case in Section 5.3, by formulating the following Lagrangian

$$\begin{aligned} \mathcal{L}(\Omega, \boldsymbol{u}, \boldsymbol{\lambda}) = \mathcal{J}(\Omega, \boldsymbol{u}) &- \int_\Omega \boldsymbol{\lambda} \cdot [\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) + \boldsymbol{f}] \, \mathrm{d}\,\Omega \\ &- \int_{\Gamma_D} \boldsymbol{u} \cdot (\boldsymbol{C} : \nabla\boldsymbol{\lambda})\boldsymbol{n} \, \mathrm{d}\,\Gamma - \int_{\Gamma_N} \boldsymbol{\lambda} \cdot [\bar{\boldsymbol{t}} - \boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n}] \, \mathrm{d}\,\Gamma \end{aligned} \tag{6.6}$$

The Lagrangian $\mathcal{L}(\Omega, \boldsymbol{u}, \boldsymbol{\lambda})$ depends on the unknown domain shape Ω, the displacement field $\boldsymbol{u}$ and the Lagrange parameter $\boldsymbol{\lambda}$. For subsequent derivations, the second term is rewritten with the divergence theorem to obtain

$$\begin{aligned} \mathcal{L}(\Omega, \boldsymbol{u}, \boldsymbol{\lambda}) = \mathcal{J}(\Omega, \boldsymbol{u}) &+ \int_\Omega \nabla\boldsymbol{\lambda} : \boldsymbol{\sigma}(\boldsymbol{u}) \, \mathrm{d}\,\Omega - \int_\Omega \boldsymbol{\lambda} \cdot \boldsymbol{f} \, \mathrm{d}\,\Omega \\ &- \int_{\Gamma_D} \boldsymbol{u} \cdot (\boldsymbol{C} : \nabla\boldsymbol{\lambda})\boldsymbol{n} + \boldsymbol{\lambda} \cdot \boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} \, \mathrm{d}\,\Gamma - \int_{\Gamma_N} \boldsymbol{\lambda} \cdot \bar{\boldsymbol{t}} \, \mathrm{d}\,\Gamma \end{aligned} \tag{6.7}$$

The stationary condition for the Lagrangian, i.e. $\delta\mathcal{L}(\Omega, \boldsymbol{u}, \boldsymbol{\lambda}) = 0$, yields the complete set of shape optimisation equations. For example, the adjoint problem for compliance 6.4

minimisation is given by considering the variation of the Lagrangian with respect to the displacements $\boldsymbol{u}$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\delta\boldsymbol{u} + \frac{\partial \mathcal{L}}{\partial(\nabla \boldsymbol{u})}\delta(\nabla \boldsymbol{u}) = \frac{\partial \mathcal{J}}{\partial \boldsymbol{u}}\delta u + \int_\Omega \nabla\boldsymbol{\lambda} : \boldsymbol{C} : \delta(\nabla \boldsymbol{u}) \,\mathrm{d}\,\Omega \\ - \int_{\Gamma_D} \delta\boldsymbol{u} \cdot (\boldsymbol{C} : \nabla\boldsymbol{\lambda})\boldsymbol{n} + \boldsymbol{\lambda} \cdot (\boldsymbol{C} : \delta(\nabla \boldsymbol{u}))\,\boldsymbol{n} \,\mathrm{d}\,\Gamma = 0\end{aligned} \tag{6.8}$$

After introducing the cost function (6.4) and reformulating the domain term with the divergence theorem

$$\begin{aligned}\int_\Omega \boldsymbol{f} \cdot \delta\boldsymbol{u} \,\mathrm{d}\,\Omega + \int_{\Gamma_N} \bar{\boldsymbol{t}} \cdot \delta\boldsymbol{u} \,\mathrm{d}\,\Gamma - \int_\Omega \delta\boldsymbol{u} \cdot (\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{\lambda})) \,\mathrm{d}\,\Omega \\ - \int_{\Gamma_D} \boldsymbol{\lambda} \cdot (\boldsymbol{C} : \nabla(\delta\boldsymbol{u}))\,\boldsymbol{n} \,\mathrm{d}\,\Gamma + \int_{\Gamma_N} \delta\boldsymbol{u} \cdot \boldsymbol{\sigma}(\boldsymbol{\lambda})\boldsymbol{n} \,\mathrm{d}\,\Gamma = 0\,.\end{aligned} \tag{6.9}$$

The corresponding boundary value problem, referred to as the adjoint problem, reads

$$-\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{\lambda}) = -\boldsymbol{f} \quad \text{in } \Omega \tag{6.10a}$$

$$\boldsymbol{\lambda} = \boldsymbol{0} \quad \text{on } \Gamma_D \tag{6.10b}$$

$$\boldsymbol{\sigma}(\boldsymbol{\lambda}) \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N \tag{6.10c}$$

It is clear that $\boldsymbol{\lambda} = -\boldsymbol{u}$ is the solution of the adjoint problem. A great advantage of the use of the Lagrangian as is the identity

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\Omega}(\Omega, \boldsymbol{u}(\boldsymbol{x}))\delta\boldsymbol{v} = \frac{\partial \mathcal{L}}{\partial \Omega}(\Omega, \boldsymbol{u}(\boldsymbol{x}), \boldsymbol{\lambda})\delta\boldsymbol{v} \tag{6.11}$$

This enables the shape derivative (4.4) to be expressed as a boundary integral of the following form

$$\begin{aligned}D\mathcal{J}(\boldsymbol{x}, u(\boldsymbol{x}))[\delta\boldsymbol{v}] &= \int_\Omega f(\boldsymbol{u}, \boldsymbol{\lambda}) \,(\nabla \cdot \delta\boldsymbol{v}) \,\mathrm{d}\,\Omega \\ &= \int_\Gamma f(\boldsymbol{u}, \boldsymbol{\lambda})(\delta\boldsymbol{v} \cdot \boldsymbol{n}) \,\mathrm{d}\,\Gamma\,,\end{aligned} \tag{6.12}$$

Without loss of generality, it can be assumed that some boundary variations are not relevant in practical shape optimisation. In solid mechanics, the boundary variations

are usually of the form

$$\begin{aligned} \delta\hat{\boldsymbol{v}} &= \boldsymbol{0} \quad \text{on} \;\; \Gamma_D\,, \\ \delta\hat{\boldsymbol{v}} &= \boldsymbol{0} \quad \text{on} \;\; \Gamma_N \;\; \text{with} \;\; \boldsymbol{\sigma n} = \bar{\boldsymbol{t}}\,, \\ \delta\hat{\boldsymbol{v}} &\neq \boldsymbol{0} \quad \text{on} \;\; \Gamma_N \;\; \text{with} \;\; \boldsymbol{\sigma n} = \boldsymbol{0}\,. \end{aligned} \tag{6.13}$$

This means that only parts of the boundary Γ_N with no traction are free to move during the shape optimisation. In this context, the variation of the Lagrangian (6.7) in the direction $\delta\hat{\boldsymbol{v}}$ with structural compliance (6.4) as the cost function reads

$$\frac{\partial \mathcal{L}}{\partial \Omega}\delta\hat{\boldsymbol{v}} = \int_{\Gamma_N} \left(2\boldsymbol{u}\cdot\boldsymbol{f} - \nabla\boldsymbol{u} : \boldsymbol{\sigma}(\boldsymbol{u})\right)\left(\delta\hat{\boldsymbol{v}}\cdot\boldsymbol{n}\right)\,\mathrm{d}\,\Gamma \tag{6.14}$$

It is worth emphasising without restricting $\delta\hat{\boldsymbol{v}}$ as stated in (6.13) the variation of the Lagrangian would contain several more terms. During the iterative shape optimisation the shape derivative (6.14) is used as gradient information. In order to achieve maximum decrease in the objective function the boundary perturbation has to be chosen in the direction

$$\delta\hat{\boldsymbol{v}} = -\left(2\boldsymbol{u}\cdot\boldsymbol{f} - \nabla\boldsymbol{u} : \boldsymbol{\sigma}(\boldsymbol{u})\right)\,. \tag{6.15}$$

6.1.2. Immersed methods for optimisation

By requiring only a definition of the domain boundary, immersed methods are ideally suited for shape optimisation of solids. Examples of immersed methods for shape optimisation in literature include the global finite difference shape derivatives and local refinement of the background grid in [53] and evolutionary strategies in [52].

In the present work, a B-spline based immersed finite element method introduced in Rüberg and Cirak [110] is used for solving the linear elastic boundary value problem (6.1) and computing the shape derivative (6.12). Several important features of the method are explained in Appendix E.1, see also Rüberg and Cirak [109, 110].

6.2. Linear elastic shape optimisation examples

Three examples are presented to demonstrate the functioning of the proposed multiresolution framework for shape optimisation of two- and three-dimensional solids in combination with immersed methods (Section 6.1.2). The problem domain is created by immersing a curve or surface in a background grid. The immersed curve or surface now represents either an internal (cavity) or external boundary of the problem domain. The mechanical problem is then solved using the proxy domain in the background grid using an immersed method with the B-spline degree set to $p = 2$. This leads to continuous stresses and shape derivatives for the discretised problem. The optimisation problem consists of determining the optimum shape of the immersed curve or surface such that the compliance (6.4) is minimised. The MMA algorithm is used to solve the minimisation problem, the input to the algorithm is as described in Section 5.4. Essentially this consists of the cost function $\mathcal{J}(\boldsymbol{x}^{\ell_c}, \boldsymbol{u}(\boldsymbol{x}^{\ell_c}))$ evaluated in the fine resolution and the position vectors $\boldsymbol{x}_i^{\ell_o}$ and the gradients $\boldsymbol{g}_i^{\ell_o}$ for each coarse geometry node in the immersed domain Ω. In addition suitable geometric bounds for the design variables are provided to the MMA algorithm.

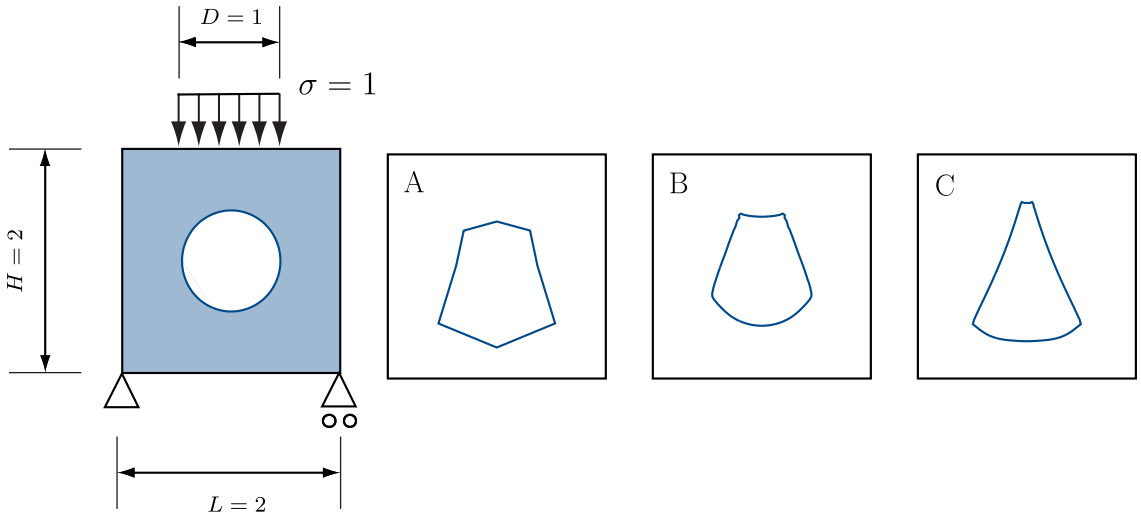
6.2.1. Two-dimensional plate with hole

This introductory example is aimed at highlighting the advantages of multiresolution optimisation. The problem consists of a square plate of dimensions 2×2 with an initially circular hole of unit diameter as shown in Figure 6.2a. The square is loaded with an in-plane line load of unit width on the top edge while the Young's modulus and Poisson's ratio are 100 and 0.4, respectively. The shape of the hole is to be optimised so that the compliance of the plate is minimised, or the stiffness is maximised. This results in a cavity with vanishing diameter unless the area of the hole is kept constant. The following area constraint is used to this end;

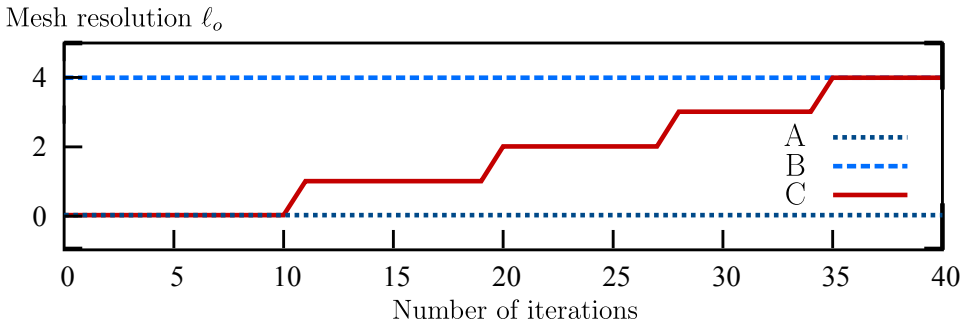
$$A_0 - A_t \leq 0 \tag{6.16}$$

where A_o and A_t denote the initial and current area of the hole, respectively. For the polygon mesh describing the hole geometry, the area is given by

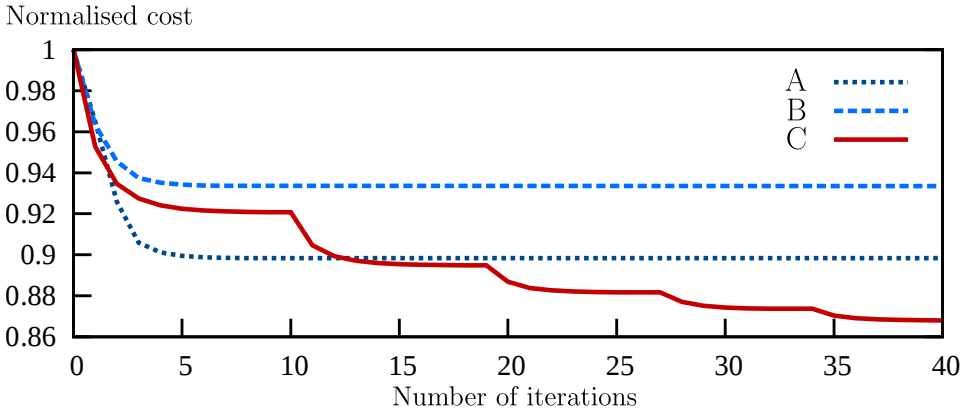
$$A_t = \sum \frac{1}{2} l_i \boldsymbol{n}_i^{\ell_c} \cdot \boldsymbol{x}_i^{\ell_c} \tag{6.17}$$



(a) Problem description(left) and hole shapes obtained for three different optimisation scenarios.



(b) Change of mesh resolution during optimisation.



(c) Convergence of normalised cost.

Figure 6.2.: Simply supported plate with a hole. The shape of the central hole is optimised using a single coarse resolution in **A**, a single fine resolution in **B** and multiple resolutions in **C**. In case **A**, both optimisation and computations are done at $\ell_o = \ell_c = 0$, whereas in case **B** both resolutions are fixed at $\ell_o = \ell_c = 4$. In case **C**, the optimisation resolution successively increased from 0 to 4 while the computational resolution is maintained at $\ell_c = 4$.

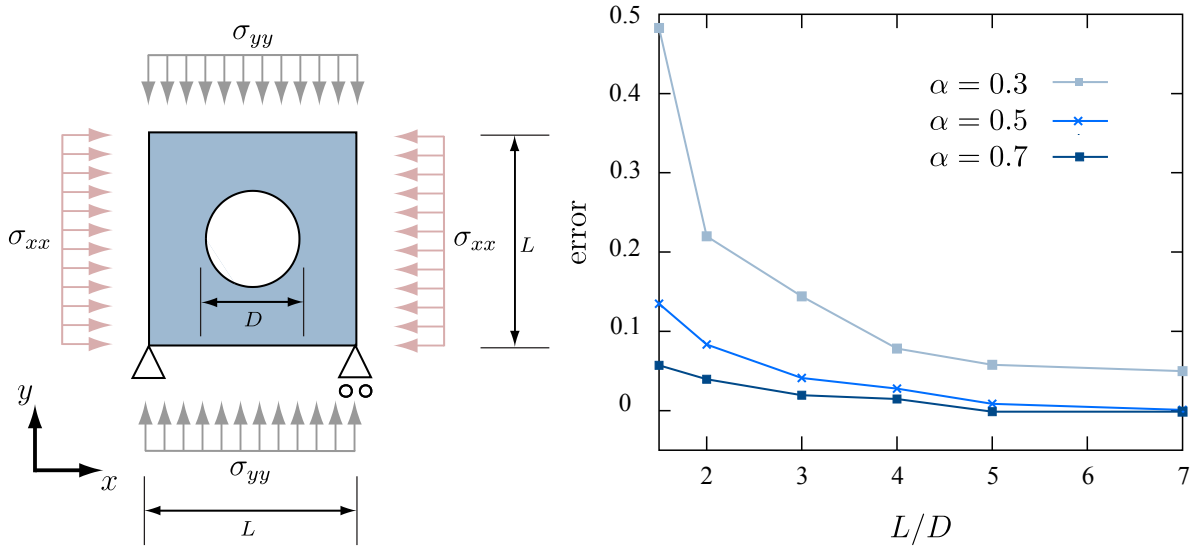
where $\boldsymbol{n}_i^{\ell_c}$ and l_i are the normal and element length at fine resolution node i, respectively. These quantities are obtained by averaging element quantities at nodes. The derivative of the constraint with respect to each fine resolution nodes is given by

$$\frac{\mathrm{d}A_t}{\mathrm{d}\boldsymbol{x}_i} = \frac{1}{2} l_i \boldsymbol{n}_i \tag{6.18}$$

Multiresolution geometry description: uniform cubic B-spline curves
A uniform cubic B-spline based multiresolution framework, the 2D equivalent of Catmull-Clark surfaces, is used in this example where initially the hole geometry is specified with 8 control points. This represents the initial geometry of the optimised shape at resolution $\ell_o = 0$. Three different cases are studied. In case **A**, both optimisation and computations are done at $\ell_o = \ell_c = 0$. Subsequently, the initial control polygon is subdivided 4 times to obtain a fine resolution mesh with 128 control points. This fine mesh is used for both analysis and optimisation in case **B**, i.e. $\ell_o = \ell_c = 4$. In case **C**, the optimisation resolution ℓ_o is successively increased from 0 to 4 while the computations are done in $\ell_c = 4$. Figure 6.2b shows the variation of optimisation resolution ℓ_o over the number of optimisation iterations. The background grid contains 100×100 cells in all case. The history of the objective function reduction for each case is shown in Figure 6.2b. It is evident that the fixed fine resolution **B** achieves the least cost reduction while the multiresolution case **C** obtains the lowest optimum. The presence of sharp corners in optimised geometry suggest an ill conditioned problem (similar to $\sigma_{xx}\sigma_{yy} < 0$ case in example 6.2.2) with several local minima. This explains the poor performance of the fine resolution which is more likely to converge to a local minima, whereas the coarse and the multiresolution cases **A** and **C** are restricted from doing this due to lack of degrees of freedom initially.

6.2.2. Optimal shapes of a cavity in an elastic domain

This classical example studies the shape of a cavity in an elastic plate with a uniform applied stress field. The problem setup is shown in Figure 6.3a. In the two-dimensional case, analytical results by Kristensen and Madson [84] show that the optimum cavity shape is an ellipse with the semi-axis ratio $r_x/r_y = \sigma_{xx}/\sigma_{yy}$. Similarly, in the three-dimensional case the optimum cavity shape is an ellipsoid with three different semi-axis ratios; $r_x/r_y = \sigma_{xx}/\sigma_{yy}$, $r_x/r_y = \sigma_{xx}/\sigma_{zz}$ and $r_z/r_y = \sigma_{zz}/\sigma_{yy}$. Numerical optimisation



(a) Problem description

(b) Convergence of optimum ellipse semi-axis ratio. The error is the deviation from the theoretical optimum ratio $\alpha = \sigma_{xx}/\sigma_{yy}$, i.e. error=$\alpha - r_x/r_y$.

Figure 6.3.: Optimal shape of cavity in an elastic plate.

studies on this problem are available in, amongst others, Noboru et al. [97] using grid adaptation, Norato et al. [98] where an immersed method is used with a filter on volume fraction, Cervera and Trevelyan [24] using boundary element methods and evolutionary optimisation strategies.

In all cases considered, the Young's modulus is 100 and the Poisson's ratio is 0.4. A plane strain state is assumed in 2D computations. An area constraint (6.16) is used to prevent the hole from shrinking in the 2D case. Similarly in a 3D setting, a volume constraint is required to prevent the cavity from shrinking

$$V_0 - V_t \leq 0 \tag{6.19}$$

where V_o and V_t denote the initial and current volume of the cavity respectively. The latter is given by

$$V_t = \sum \frac{1}{3} A_i \boldsymbol{n}_i^{\ell_c} \cdot \boldsymbol{x}_i^{\ell_c} \tag{6.20}$$

where $\boldsymbol{n}_i^{\ell_c}$ and A_i are the normal and element area at fine resolution node i, respectively. As in the 2D case, nodal quantities are obtained by averaging element quantities at nodes. The derivative of the constraint with respect to each fine resolution nodes is

given by

$$\frac{\mathrm{d}V_t}{\mathrm{d}\boldsymbol{x}_i} = \frac{1}{3} A_i \boldsymbol{n}_i \tag{6.21}$$

Multiresolution geometry description: uniform cubic B-spline curves
An coarse resolution model with 8 control points is used as the initial hole geometry at $\ell_o = 0$. This is refined up to $\ell_o = 3$ during the multiresolution optimisation process. All computations are done at a fine resolution $\ell_c = 3$.

First the effects of having a finite domain, as opposed to the infinite domain in the analytical setting, is studied. To this end, the L/D ratio of the plate is changed while keeping a constant cell density of 25×25 cells/unit area in the background grid representing the plate. The diameter of the hole at the start is fixed at $D = 1$ and the plate size is varied $1.5 \leq L \leq 7$. Figure 6.3b shows the error in semi-axis ratio of the obtained ellipses for different stress ratios $\alpha = \sigma_{xx}/\sigma_{yy}$. During analysis, α values are specified by setting $\sigma_{yy} = 10$ and $\sigma_{xx} = \alpha\sigma_{yy}$. Convergence to the correct semi-axis ratio is observed for stress ratios $\alpha = 0.5$ and $\alpha = 0.7$ after increasing the domain size. However convergence to the correct value is not achieved using the current cell size for the stress ratio $\alpha = 0.3$, for which the discretisation error is too large near the semi-major axis of the ellipsoid.

Next, the grid dimensions and cell density are kept fixed at $D = 1$, $L/D = 4$ and 100×100 cells/unit area respectively to focus on variation of optimal shapes for different values of $\alpha = \sigma_{xx}/\sigma_{yy}$. As before, α values are specified during analysis by setting $\sigma_{yy} = 10$ and $\sigma_{xx} = \alpha\sigma_{yy}$ with $\alpha \in \{-1.0, -0.7, -0.5, -0.3, -0.1, 0.1, 0.3, 0.5, 0.7, 1.0\}$. Cherkaev et al. [26] show that when $\sigma_{xx}\sigma_{yy} < 0$, simply connected shapes stop being optimal. Essentially, having more than one cavity becomes more optimal in this case. In the present computations, the formation of more cavities is prevented by adding a term to the cost function that penalises the increase of the hole's perimeter

$$\mathcal{J}(\boldsymbol{x}^{\ell_c}, u(\boldsymbol{x}^{\ell_c})) \leftarrow \mathcal{J}(\boldsymbol{x}^{\ell_c}, u(\boldsymbol{x}^{\ell_c})) + \rho_L \sum (l_i)^2 \tag{6.22}$$

where l_i is averaged element length at node i and ρ_L is a scalar penalty parameter (determined via numerical experimentation). The complete range of optimum cavity shapes can now be computed as shown in Figure 6.4. Note that as $|\alpha| \to 0$, the discretisation error starts dominating and the deviation from the optimum semi-axis ratio increases as observed in the previous study.

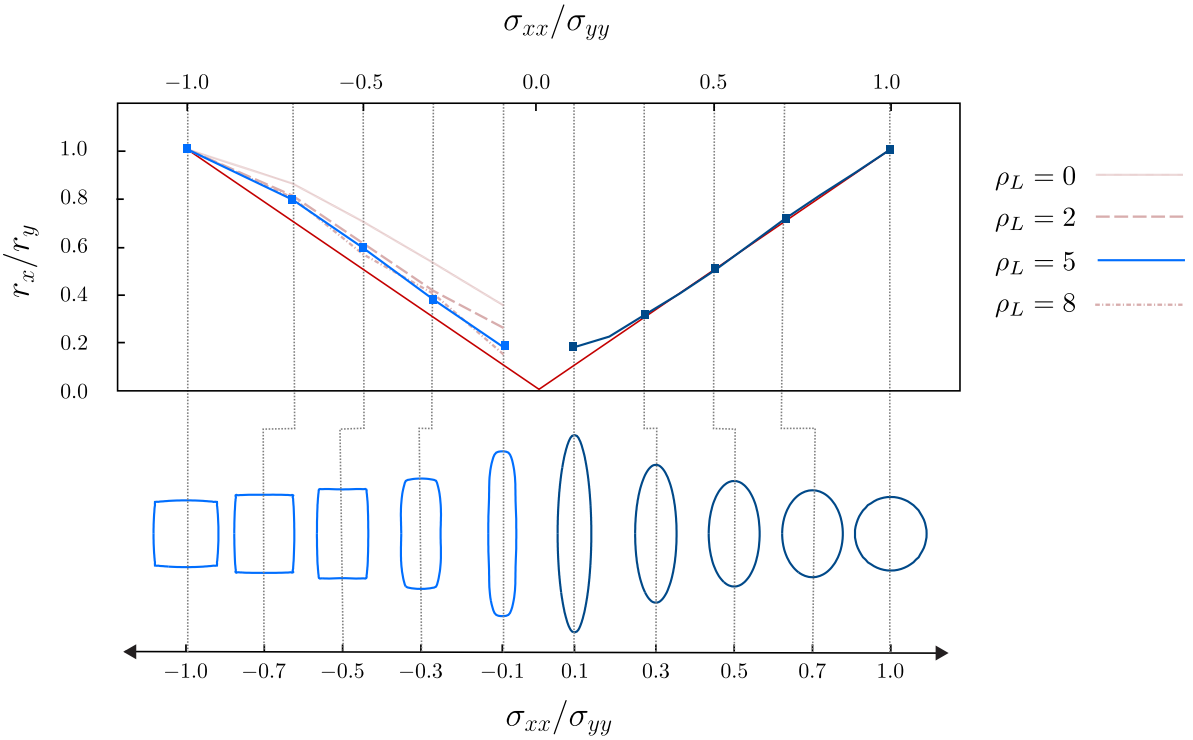


Figure 6.4.: Optimum shapes and semi-axis ratios of cavities in an elastic plate. The theoretical semi-axis ratio is shown in red. The multiple curves for $\sigma_x\sigma_y < 0$ are computed using different penalty values ρ_L for the perimeter (6.22). The corresponding optimum shapes (bottom row) are for $\rho_L = 5$. No penalty is applied when $\sigma_x\sigma_y > 0$.

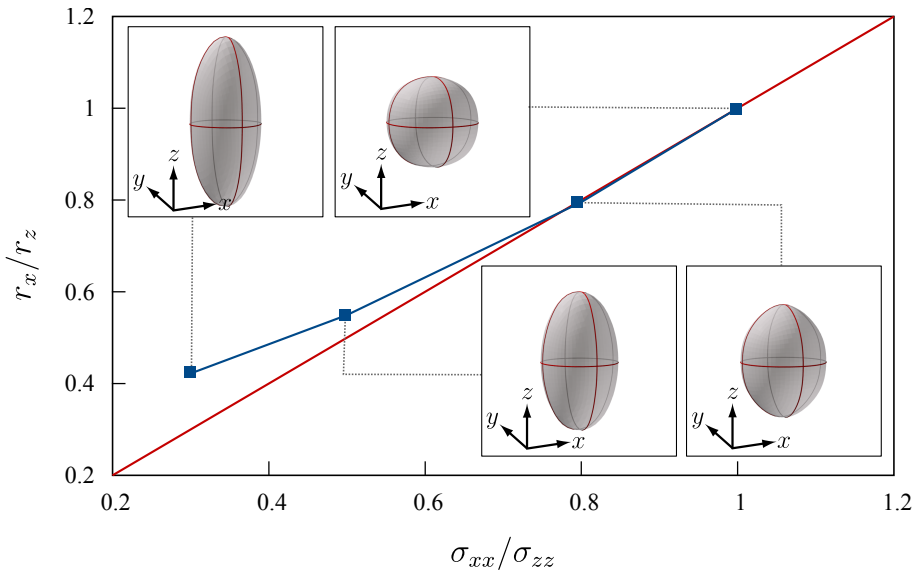


Figure 6.5.: Optimum shapes and semi-axis ratios of cavity in an infinite domain. Note that r_x, r_z denote the length of ellipsoid semi-axis in x and z directions respectively. The theoretical semi-axis ratio is shown in red.

Multiresolution gometry description: Catmull-Clark subdivison

The 3D equivalent of the previous 2D studies is made by immersing a spherical hole inside a 3D domain. A quadrilateral subdivision mesh with 26 control points is used to represent the initially spherical cavity of diameter $D = 1$. The control points of this mesh are used as the initial design variable at $\ell_o = 0$. During multiresolution optimisation, the initial mesh is refined up to 3 times, i.e. $\ell_o = 0 \rightarrow 3$.

First the cavity shape is optimised for different stress ratios $\sigma_{xx}/\sigma_{zz} \in \{0.3, 0.5, 0.8, 1.0\}$ while $\sigma_{xx} = \sigma_{yy}$ is kept constant. In computations, $\sigma_{zz} = 10$ is chosen and σ_{xx} and σ_{zz} are changed accordingly. The 3D elastic domain is represented by a background grid in the shape of cube. The grid dimensions are maintained at $L = 4$ with cell density of $(20 \times 20 \times 20/\text{unit volume})$. Figure 6.5 shows the final semi-axis ratio r_x/r_z of the resulting ellipsoids. In comparison to the equivalent 2D results in Figure 6.4, a larger deviation from the theoretical ratio is observed. Note that although the same $L/D = 4$ ratio is set in both cases, the discretisation error is much larger in the 3D case as a result of the coarse grid used.

Similarly, two stress ratios can be varied at once resulting in optimised shapes shown in Figure 6.6. Here, σ_{xx}/σ_{yy} and σ_{zz}/σ_{yy} are both varied while keeping $\sigma_{yy} = 3$. A constant cell density of $(5 \times 5 \times 5$ cells/unit area) is used while the domain size $L_x \times L_y \times L_z$ is

varied in a proportional manner to the applied stresses as follows;

$$L_x = 3|\sigma_{xx}/\sigma_{yy}|, \quad L_y = 3, \quad L_z = 3|\sigma_{zz}/\sigma_{yy}|$$

When negative stress ratios are present (Figure 6.6b), a penalty method similar to 6.22 is used to prevent formation of additional cavities by penalising the increase of surface area.

$$\mathcal{J}(\boldsymbol{x}^{\ell_c}, u(\boldsymbol{x}^{\ell_c})) \leftarrow \mathcal{J}(\boldsymbol{x}^{\ell_c}, u(\boldsymbol{x}^{\ell_c})) + \rho_A \sum (A_e)^2 \tag{6.23}$$

where A_e denotes the surface area of fine resolution elements. In Figure 6.6a, the positive stress ratios applied have resulted in the optimum cavity shape being ellipsoids where the semi-axis ratios are visually proportional to the applied stress ratios. The negative stress ratios in Figure 6.6b result in optimised shapes where the cross-section in the xy plane is an ellipse while nearly rectangular cross sections are obtained in the xz, yz planes.

Mutiresolution gometry description: progressive meshes

The progressive mesh framework is now used to demonstrate an extension of the previous study, where shear stresses are applied to the domain boundaries causing the cavity to rotate. However it should be made clear that progressive meshes are not suited for optimisation problems of this nature, as use of details tend to preserve the original geometry to some degree. This is more conspicuous during coarse resolution edits where details are long. However the details are shorter during finer resolution edits enabling convergence to the correct optimum shape to some degree.

In contrast to the previous setting, if shear stresses are applied to the domain, the resulting optimised ellipsoids are rotated according to the eigenvectors of the stress tensor. For example, the following stress conditions result in the optimum shape being an ellipsoid rotated by 45^0;

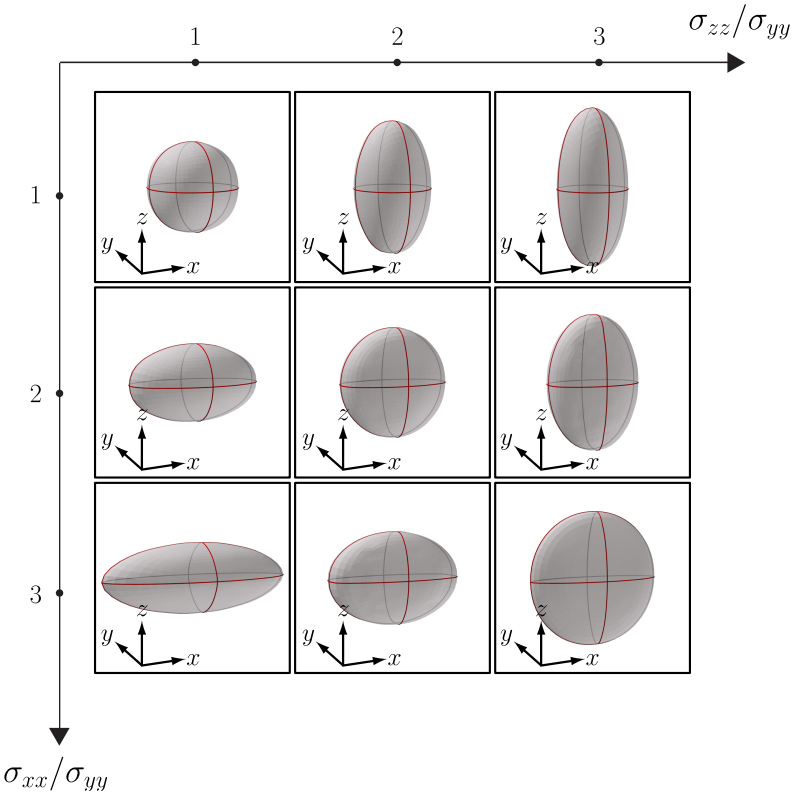
case **A**

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

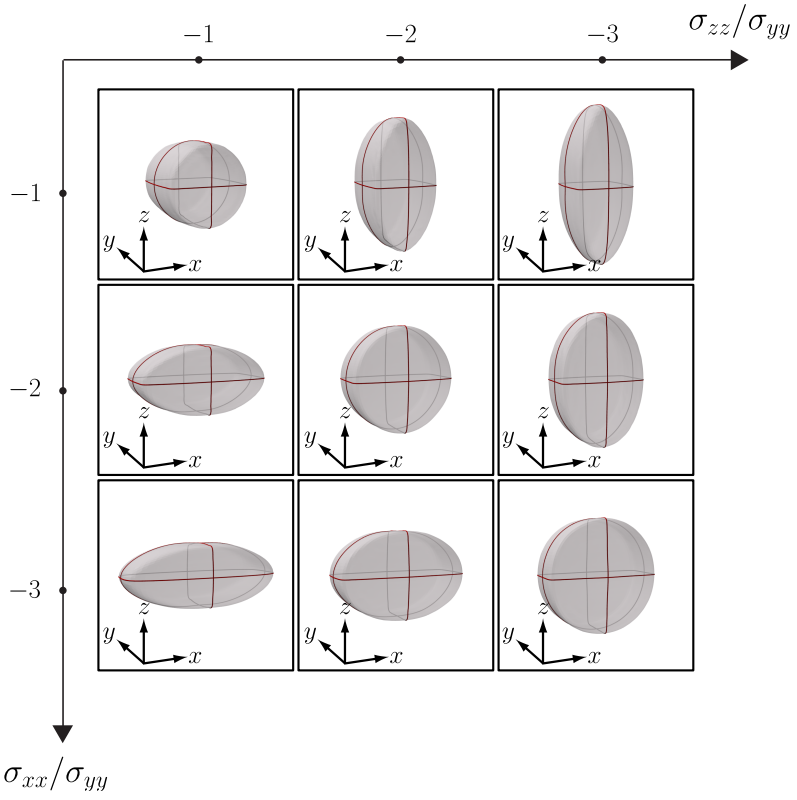
case **B**

$$\boldsymbol{\sigma} = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

The principal stresses of $2, 2, 4$ and $2, 4, 4$ given by the eigenvalues, are now rotated with

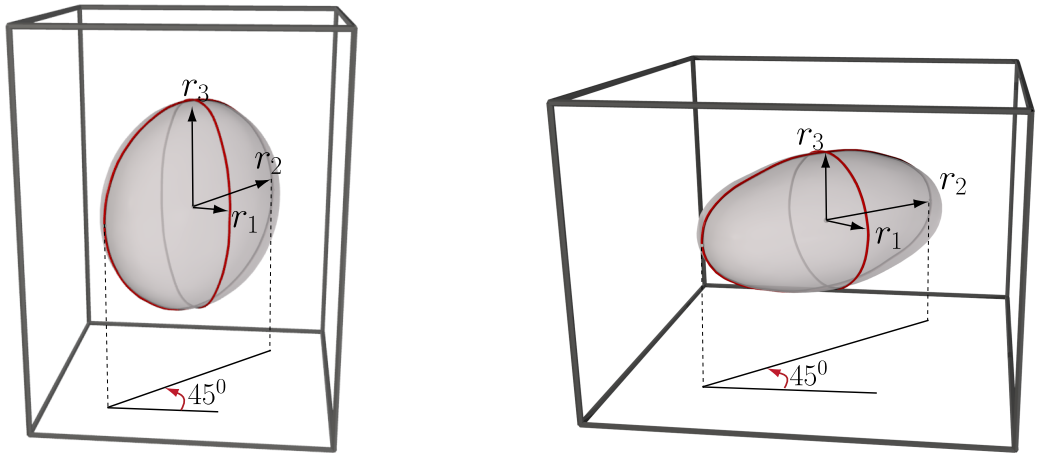


(a) Positive stress ratios; $\sigma_{xx}/\sigma_{zz},\ \sigma_{yy}/\sigma_{zz} \geq 0$



(b) Negative stress ratios; $\sigma_{xx}/\sigma_{zz},\ \sigma_{yy}/\sigma_{zz} \leq 0$

Figure 6.6.: Comparison of optimum cavity shapes for different stress ratios σ_{xx}/σ_{zz} and σ_{yy}/σ_{zz}.



(a) Optimised geometry for case **A**.

(b) Optimised geometry for case **B**.

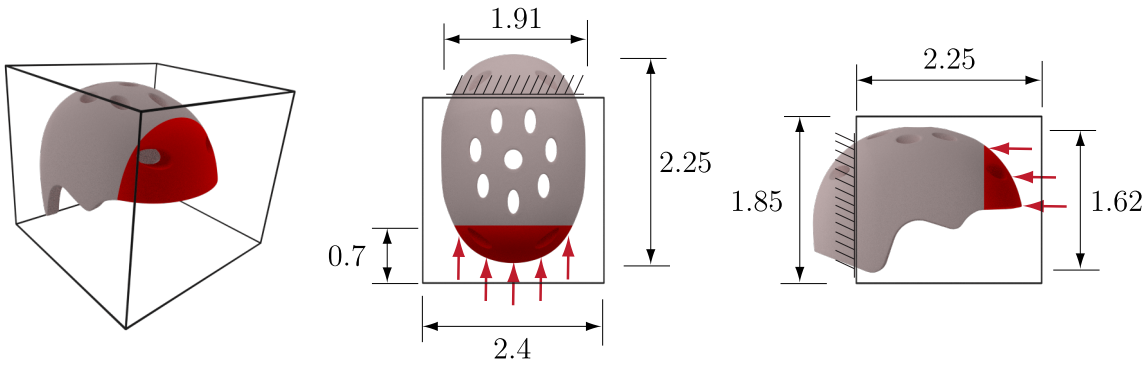
Figure 6.7.: Optimum cavity shape for stress conditions involving shear. An initially spherical cavity is optimised using progressive meshes. The outline of the background grid representing the domain is shown in black.

two of the principal directions at an angle of 45^0 to the basis vectors.

A mesh $\mathbf{x}^{24568}$ with 24568 vertices representing a sphere of diameter 1 is used as the initial geometry of the cavity. This fine resolution mesh is immersed in a grid of, $3\times4\times3$ in case **A** and $3 \times 4 \times 3$ in case **B**. The cell density is a constant $5 \times 5 \times 5$ cells/unit volume in both cases. During optimisation, the geometry is decimated to obtain the optimisation resolution $\ell_o \in \{245, 798, 1351, 2457\}$. Figure 6.7 shows the final optimised shapes in the form of rotated ellipsoids as expected. The obtained semi-axis ratios are 1: 1.01: 1.96 for case **A** and 1: 1.81: 1.86 for case **B**. The error with respect to the theoretical ratios 1: 1: 2 and 1: 2: 2 is comparable with that of subdivision given the coarse grid used. However, it should be noted that the rotation angle is recovered precisely with 45^0 in each case.

6.2.3. Shape optimisation of skate helmet

The progressive mesh framework is ideally suited to making relative small shape changes to existing detailed designs. Such a situation is demonstrated in this example where the outer shape of a skate helmet is optimised to improve its overall stiffness in the longitudinal direction. The initial geometry of the helmet is given as a fine resolution surface mesh with 30538 vertices. This is partially immersed in a background grid of dimensions $2.25 \times 2.4 \times 1.85$ as depicted in Figure 6.8a. The grid contains $30 \times 30 \times 30$



(a) 3D view (b) Top view (c) Side view

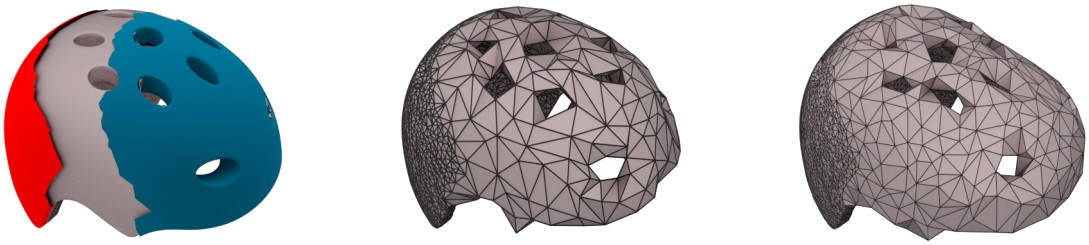
Figure 6.8.: Skate helmet problem setup. Original CAD geometry obtained from http://www.grabcad.com. The fine resolution skate helmet mesh of dimensions $2.25 \times 1.91 \times 1.62$ is partially immersed in a background grid. A unit pressure is applied to the front of the helmet indicated in red.

cells. The optimisation problem is created by applying a longitudinal pressure on a region at the front of the helmet and applying a rigid support at the back as shown in Figures 6.8b and 6.8c. The Young's modulus and Poisson's ratio are 100 and 0.4, respectively. A volume constraint (6.19) is used to keep the volume constant during optimisation.

Note that these conditions are only used for creating an illustrative problem, the formal procedure for testing of such helmets is governed by the standard BSEN 1078 [1]. The latter involves impact testing, a numerical study of which is available in Mills and Gilchrist [94].

Mutiresolution gometry description: progressive meshes

During optimisation, only the front portion of the outer surface needs to be optimised while the back and inner surface, the latter touching the wearer's head, requires no shape changes. The shape change must be such that the ventilation holes are preserved. These design constraints require additional control during decimation. Specifically, a vertex distribution quadric (3.5) with a high penalty factor is added to all vertices in regions where no geometry changes are permitted. This results in graded coarse resolution shown in Figures 6.9b and 6.9c. Additionally only coarse resolution nodes in the front of the helmet (the blue region in Figure 6.9a) are used as active design variables. This process results in a smooth transition of shape changes between the front and the rest of the helmet. Such smooth transition cannot be achieved if a binary oracle is used to



(a) Nodes in the red area are penalised during decimation. The blue area denotes the active optimisation region.

(b) Initial coarse resolution $\mathbf{x}^{2352}$.

(c) Final coarse resolution $\mathbf{x}^{4026}$.

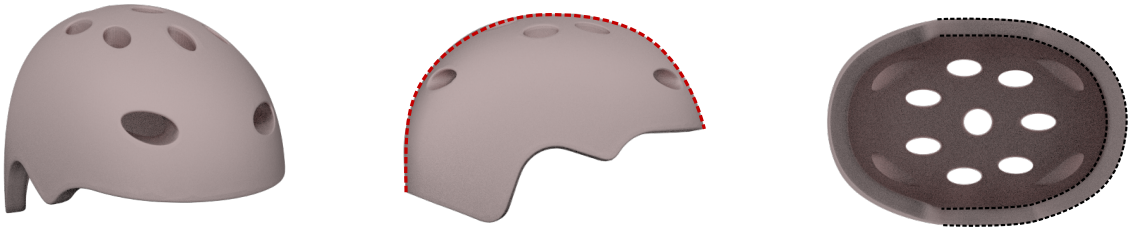
Figure 6.9.: Skate helmet mesh decimation setup. Additional controls are needed in the decimation and optimisation process to maintain smooth transition of shape changes between the front and the rest of the helmet.

prevent any decimation in the back or inner surface.

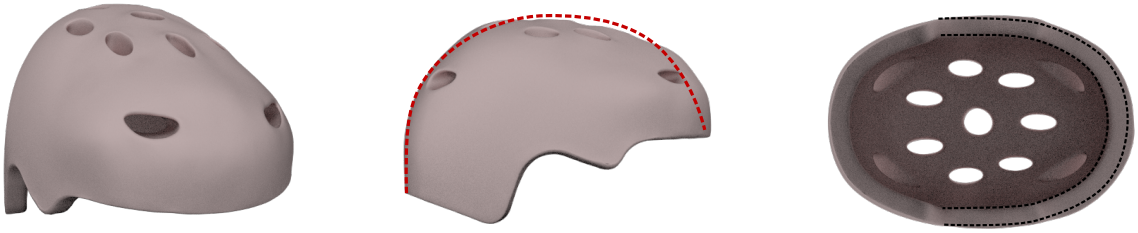
The initial mesh $\mathbf{x}^{30538}$, also used as the immersed geometry, is optimised in coarse resolutions $\ell_o \in \{2352, 2910, 3468, 4026\}$ to obtain the final optimised shape shown in Figure 6.10b. The corresponding cost reduction is $\approx 10\%$ while the reduction in the strain energy density is shown in Figure 6.11. A comparison of the geometry before and after optimisation, shown in Figure 6.10, reveals changes to the helmet profile near the front as expected. Additionally the thickness has also increased in the same region while maintaining the original inner surface and ventilation holes as required.

6.3. Linear elastic topology optimisation

Topology optimisation can be addressed using a variety of numerical methods which can be broadly classified into two groups; the material or microstructure techniques and geometrical or macrostructure techniques [47]. Informally, microstructure techniques are based on an initially porous material distribution where the constitutive model relates material density $0 \leq \rho \leq 1$ to stiffness. An optimisation problem is then solved to determine the optimum distribution of material density. Two widely used microstructure techniques are the SIMP method [10] and the homogenisation method [11, 3]. In

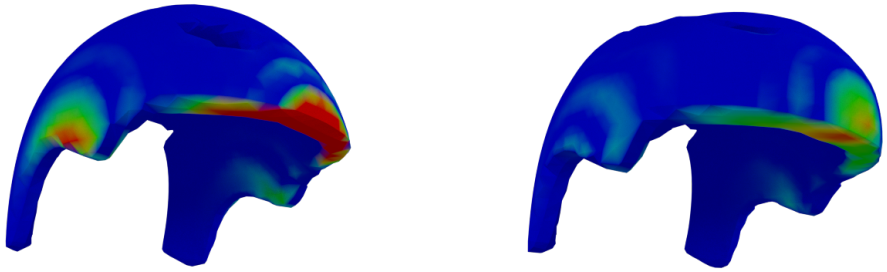


(a) Before shape optimisation.



(b) After shape optimisation.

Figure 6.10.: Comparison of skate helmet geometry before and after optimisation. Note the changes in the profile (red dashed line), and thickness (black dashed lines).



(a) Initial geometry

(b) Optimised geometry

Figure 6.11.: Strain energy density of skate helmet before and after optimisation. The geometry shown is the domain created by the immersed method.

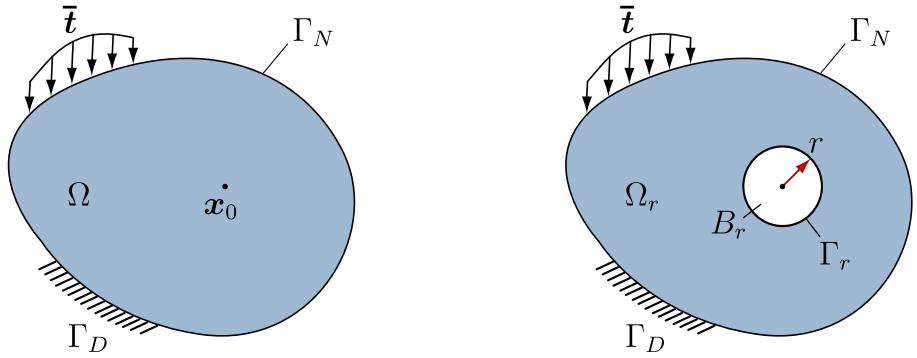
the SIMP method, intermediate material densities are penalised to favour either having $\rho = 0$ or $\rho = 1$ denoting a void or a solid respectively. In the homogenisation method intermediate material densities are treated as composites. See, amongst others, Eschenauer and Olhoff [47], Bendsøe and Sigmund [12] for a review of microstructure techniques in topology optimisation.

The macrostructure approach, treats the material as homogenous and the topology of a continuum domain is changed by adding or removing holes. This can be done in such a way that the new holes are represented as CAD surfaces, inline with the goals of integrated product development. For example, Cervera and Trevelyan [24] used a boundary element method and evolutionary concepts for topology optimisation where the new topology boundaries were represented using NURBS curves. Material is then added or removed from the domain by manipulating the NURBS control points. A formal framework for increasing the genus of a solid body is offered by the bubble method [46], which is based on iterative introduction, positioning and shape optimisation of new holes. Essentially, the optimum location to introduce a topology change is first determined followed by insertion of a small hole which is enlarged during subsequent shape optimisation using area/volume minimisation with constraints on compliance. In addition the work of Eschenauer et al. [46] who used NURBS curves to represent the bubbles, several examples of using this approach with CAD geometry exist; a parameter based approach by Schumacher [114] and a NURBS based framework by Seo et al. [117, 118]. The latter represented topology boundaries as trimming curves.

6.3.1. Topology derivative

The motivation behind the topology derivative is to predict the optimum location for placement of a new hole, i.e. optimum location for a topology change. Eschenauer et al. [46] studied the variation of an objective functional with respect to adding an infinitesimal hole. This is used to define a *characteristic function* that depends on the stress state of the domain. Subsequently, holes are inserted at the location where the characteristic function takes the minimum value. Based on this, Garreau et al. [57] presented a formal structure for evaluating the topology derivative via topological sensitivity analysis.

The topology derivative at a point gives the sensitivity of the cost function when a small hole is created at that point. Let B_r be a hole of radius r centred in $\boldsymbol{x}_0 \in \Omega$ with



(a) Initial domain Ω.

(b) Domain with new topology.

Figure 6.12.: Change of domain topology. A hole of radius r is inserted at $\boldsymbol{x}_0$ to change the topology of original domain from Ω to $\Omega_r = \Omega \setminus B_r$.

boundary Γ_r. The boundary value problem (6.1) in Ω (Figure 6.12a) is now defined in the new domain $\Omega_r = \Omega \setminus B_r$ (Figure 6.12b)

$$-\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}_r) = \boldsymbol{f} \quad \text{in } \Omega_r \tag{6.24a}$$

$$\boldsymbol{u}_r = \overline{\boldsymbol{u}} \quad \text{on } \Gamma_D \tag{6.24b}$$

$$\boldsymbol{t}(\boldsymbol{u}_r) = \boldsymbol{\sigma}(\boldsymbol{u}_r) \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \quad \text{on } \Gamma_N\,. \tag{6.24c}$$

$$\boldsymbol{\sigma}(\boldsymbol{u}_r) \cdot \boldsymbol{n} = 0 \quad \text{on } \Gamma_r \tag{6.24d}$$

Note the additional Neumann boundary condition for the new hole boundary. For the sake of brevity, the value of the objective function $\mathcal{J}(\Omega, \boldsymbol{u})$ (6.4) will be denoted by $\mathcal{J}(\Omega)$ henceforth. In the new domain Ω_r, the objective function can be expanded as follows

$$\mathcal{J}(\Omega_r) = \mathcal{J}(\Omega) + f(r) D_T \mathcal{J}(\boldsymbol{x}_0) + o(f(r)) \tag{6.25a}$$

$$\lim_{r \to 0} f(r) = 0, \quad f(r) > 0 \tag{6.25b}$$

where $f(r)$ is a measure of the hole size and $D_T \mathcal{J}(\boldsymbol{x}_0)$ is the topological derivative at $\boldsymbol{x}_o$ (see also Sokolowski and Zochowski [120] and Céa et al. [23]);

$$D_T \mathcal{J}(\boldsymbol{x}_0) = \lim_{r \to 0} \frac{\mathcal{J}(\Omega_r) - \mathcal{J}(\Omega)}{f(r)} \tag{6.26}$$

The domain Ω_r now varies with r and there is no continuous mapping with an inverse between Ω_r and Ω, hence the derivative (6.26) cannot be computed in a conventional way.

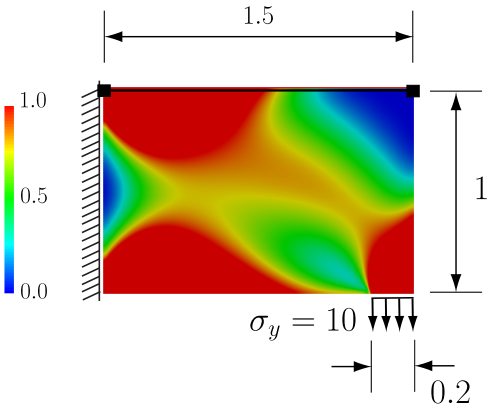
Two alternatives mathematical frameworks are used to this end, the domain truncation method [57] or the so-called topology-shape sensitivity method [99, 100, 101]. The latter relates the shape derivative to the topology derivative, and is used in the present work. Specifically, the topology derivatives given in Novotny et al. [100, 101] for minimisation of total potential energy in linear elastic boundary value problems in plane stress, plane strain and 3D are used. See appendix D.2 for more details on topology derivatives.

6.4. Topology optimisation examples

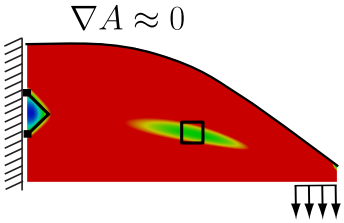
Several examples are introduced in this section to demonstrate the versatility of multiresolution optimisation in a topology optimisation context. Only the subdivision based multiresolution framework is used due to the need for new meshes during topology changes. Additionally, topology optimisation usually involves significant changes to existing topology boundaries, which is easier to accomplish with subdivision surfaces.

The problem domain is created by immersing a curve or surface, representing the boundary of the initial domain, in a background grid. Subsequently, a combination of shape and topology optimisation steps in any desired order are performed. During a topology optimisation step, the topology derivative $D_t\mathcal{J}(\boldsymbol{x})$ is computed for each cell in the background grid and a bubble is inserted at the location of minimum topology derivative. Alternatively, a group of cells with the largest topology derivative can be removed based on a volume reduction ratio. In this case the new topology boundary approximately coincides with a contour line of the topology derivative. The boundary geometry of the bubble is next optimised with shape optimisation optimisation before another bubble is inserted and the process is repeated. Each boundary in the current domain is treated as a separate multiresolution curve or surface during shape optimisation.

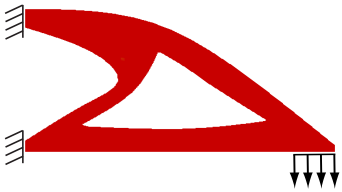
As in previous examples, the boundary value problem (6.1) is solved using the proxy domain in the background grid using the immersed method (Section 6.1.2). The shape optimisation problem consists of determining the optimum shape of the immersed curves or surfaces such that the compliance (6.4) is minimised. The MMA algorithm is used for solving the discretised optimisation problem. The input to the MMA algorithm is as described in Section 5.4. Essentially it consists of the cost function $J(\boldsymbol{x}^{\ell_c}, u(\boldsymbol{x}^{\ell_c}))$ evaluated in the fine resolution and the position vectors $\boldsymbol{x}_i^{\ell_o}$ and the gradients $\boldsymbol{g}_i^{\ell_o}$ for



(a) Problem description. The top boundary is first shape optimised with bounds on the end nodes (black squares) to prevent any horizontal shrinking.



(b) Two bubbles are introduced at locations of minimum topology derivative. Bounds restricting horizontal movement are added to the nodes indicated as black squares.



(c) Final optimised geometry.

Figure 6.13.: Topology optimisation of a cantilever truss. The colour contours show the topology derivative.

each node in the optimisation level ℓ_o. The size of the bubbles are increased during shape optimisation using area or volume constraints that are initially violated

$$g_A A_0 - A_\tau \leq 0 \tag{6.27a}$$

$$g_V V_0 - V_\tau \leq 0 \tag{6.27b}$$

where different values for $g_A \geq 1$ and $g_V \geq 1$ are used to increase the current area A_τ or volume V_τ with respect to the initial values A_o and V_o. The area and volume of the holes are computed using (6.17) and (6.20) respectively.

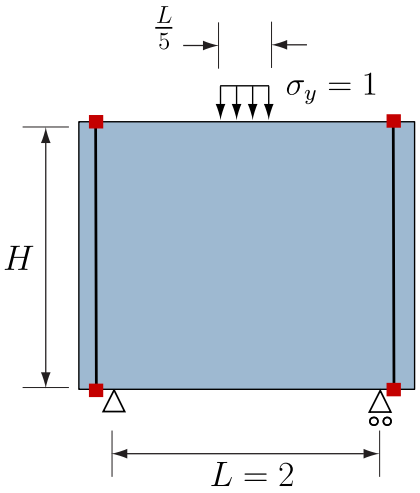


Figure 6.14.: Problem description of arch to truss transition study The two vertical boundaries are first optimised with bounds on the end nodes (red squares) to prevent any vertical movement.

6.4.1. Cantilever truss

This plane-stress topology optimisation problem, shown in Figure 6.13a, is similar to the cantilever truss problem in Eschenauer et al. [46]. However, in the present example the starting geometry is a rectangular plate, the top boundary of which is first shape optimised. The top boundary (black line in Figures 6.13a and 6.13b) is represented at the coarse resolution $\ell_o = 0$ by a cubic B-spline with 3 control points. The two end control points are tagged as corner to prevent horizontal shrinking. The fine resolution is obtained after 2 subdivisions ($\ell_c = 2$) and immersed in a background grid of 100×100 cells representing the plate. The Young's modulus and Poisson's ratio are 100 and 0.4, respectively. The shape of the curve is optimised while reducing the area of the domain using $g_A < 1$ in (6.27a).

In the second stage, two bubbles (a triangle and a square) are inserted at locations of minimum topology derivative as shown in Figure 6.13b. In the coarse resolution $\ell_0 = 0$, the bubbles contain 3 and 4 nodes which are subdivided twice to obtain the fine computation resolution $\ell_c = 2$.

6.4.2. Arch to truss transition

The aspect ratio of the domain shown in Figure 6.14 is changed during shape and topology optimisation to simulate the transition from arch to truss behaviour. The starting domain is initially rectangular and its vertical boundaries are first optimised followed by iterative positioning and optimisation of bubbles. Three cases are studied for different $H/L \in \{0.25, 0.5, 1\}$ ratios, where the background grids contain 80×200, 200×100 and 150×150 cells respectively. Young's modulus and Poisson's ratio are 100 and 0.4 respectively for all cases.

The vertical boundaries are represented at the coarse level $\ell_o = 0$ using cubic curves with 3 control points each. These are subdivided 3 times to obtain the fine resolution $\ell_c = 3$. As in the previous example, the shape of the domain is first optimised while reducing area before bubbles are introduced and optimised iteratively. Note that the bubbles contain 3 or 4 nodes at $\ell_o = 0$ and the fine resolution is chosen with $\ell_c = 3$. Figure 6.15 contains the history of intermediate topology changes and the final optimisation results for different H/L ratios indicating the expected arch behaviour for $H/L = 1$ case and truss behaviour for $H/L = 0.25$ case. The optimisation of $H/L = 1$ is a special case since having one roller support results in a small tensile region at the bottom whereas changing the support conditions to two pinned supports results in an arch (Figure 6.16).

6.4.3. Shape and topology optimisation of a table

Figure 6.17 shows the setup of a 3D shape and topology optimisation problem used as the starting geometry in this problem. Only a quarter of the model is used in the optimisation study considering symmetry with appropriate bounds and geometry tags at the planes of symmetry. The geometry is immersed in a background grid of dimensions $0.7 \times 0.7 \times 1$ and $30 \times 30 \times 30$ cells. The Young's modulus and Poisson's ratio are 100 and 0.4, respectively.

The sequence of topology and shape optimisation stages performed are shown in Figure 6.18. Unlike the previous two examples, a relatively large amount of material is removed during each topology optimisation stage. Elements with topology derivative below a certain threshold are removed in a single stage such that a clear change in topology is obtained. The initial domain is created by immersing the outer boundary

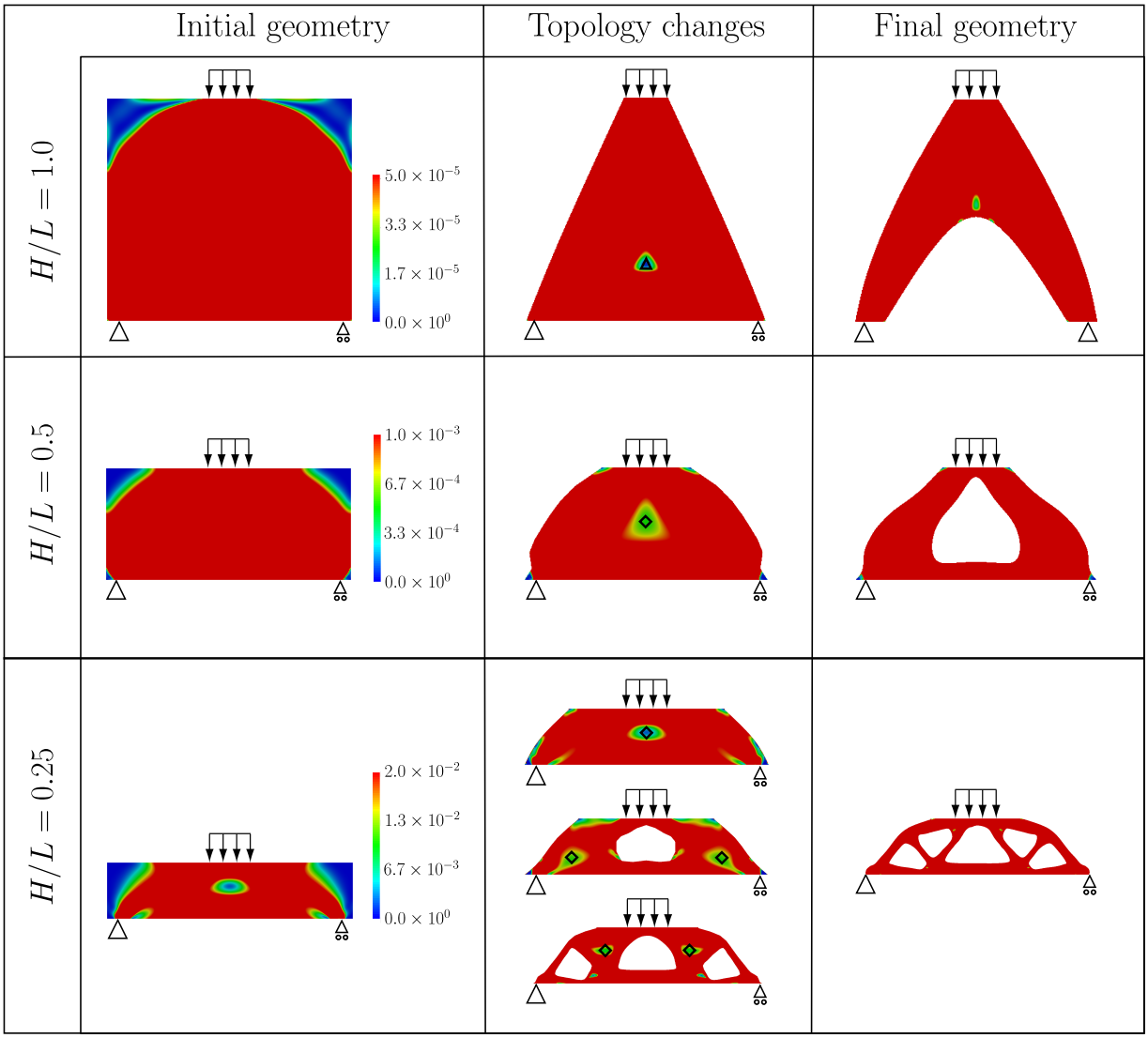
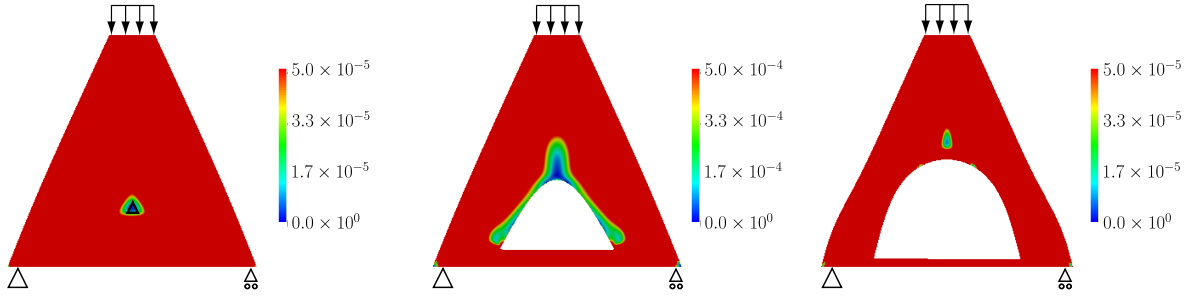
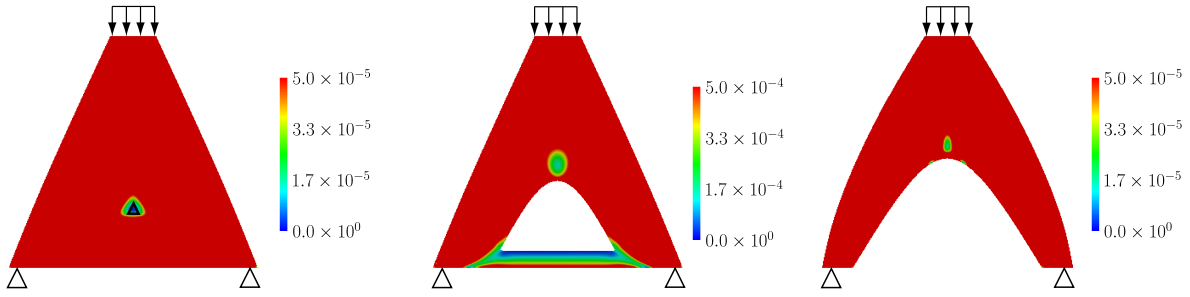


Figure 6.15.: Arch to truss transition using shape and topology optimisation. The colour contours indicate the topology derivative. After initial shape optimisation of the vertical boundaries, new bubbles are inserted at points of minimum topology derivative (middle column).



(a) Pinned support on the left and a roller support on the right.



(b) Both supports pinned. At the intermediate stage, the topology derivative is minimal below the cavity indicating removal of material to obtain an arch.

Figure 6.16.: Changing support conditions based on topology derivative for $H/L = 1$. The colour contours show the topology derivative of the initial (left), intermediate (middle) and final (right) geometries.

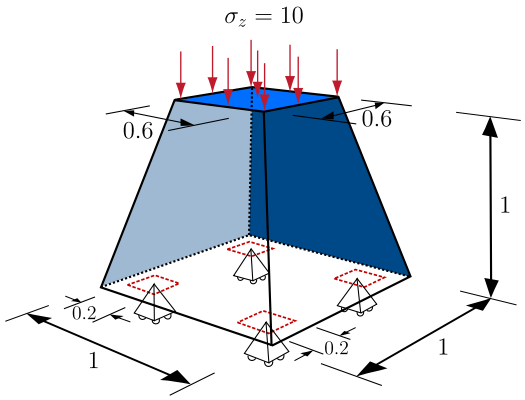
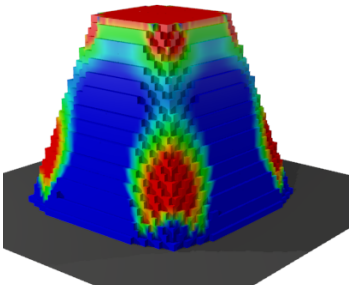
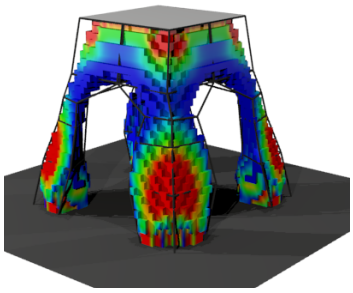


Figure 6.17.: Problem description for shape and topology optimisation of a table. Pinned supports are applied to all nodes inside regions marked as red squares.

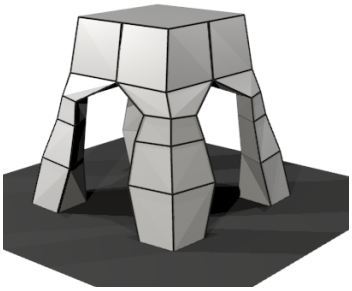
(Figure 6.17) in the background grid. The first topology optimisation step, ($A \rightarrow B$) in Figure 6.18 involves removing cells with a topology derivative $D_T\mathcal{J} \leq 0.025$. A coarse resolution subdivision surface representing the boundary of the new topology is generated next ($B \rightarrow C$). The new surface at $\ell_o = 0$ is subject to shape optimisation ($C \rightarrow D$), with the fine resolution given by $\ell_c = 2$. A volume constraint is applied to maintain constant domain volume during shape optimisation. Material with topology derivative $D_T\mathcal{J} \leq 0.04$ are removed during the second topology optimisation, stage ($E \rightarrow F$) followed by another surface generation ($F \rightarrow G$) and shape optimisation ($G \rightarrow H$) stage.



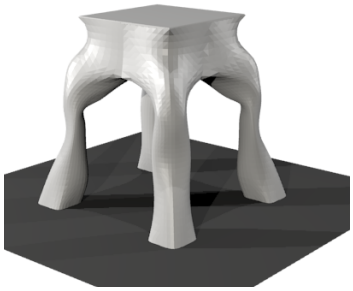
(a) Stage A: topology derivative of the initial domain.



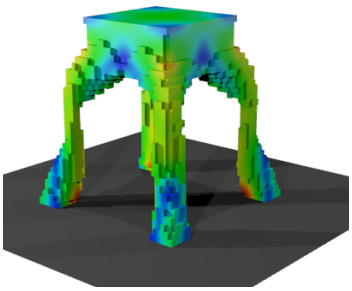
(b) Stage B: material removed to create new topology.



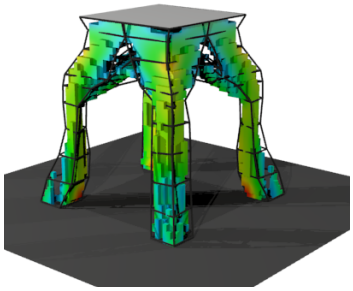
(c) Stage C: initial boundary shape.



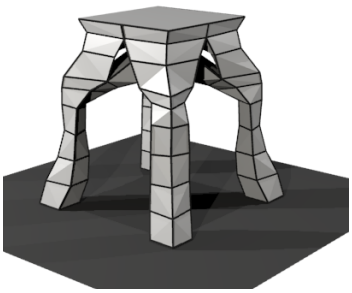
(d) Stage D: optimised boundary shape.



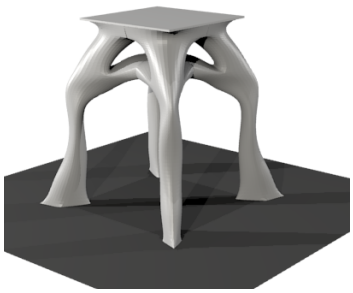
(e) Stage E: topology derivative of domain.



(f) Stage F: material removed to create new topology.

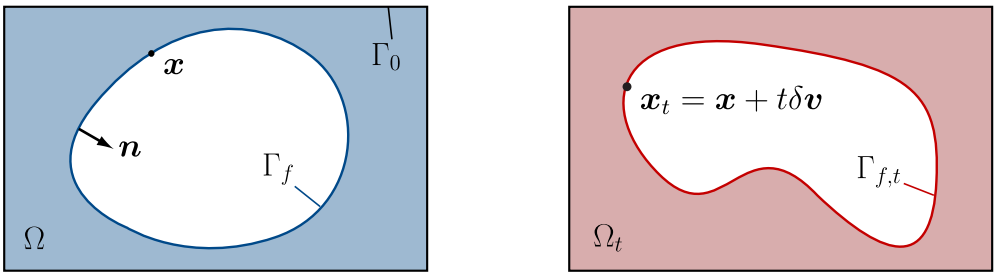


(g) Stage G: initial boundary shape.



(h) Stage H: optimised boundary shape.

Figure 6.18.: Shape and topology optimisation of a table. Topology optimisation is performed during $(A \rightarrow B)$ and $(E \rightarrow F)$, shape optimisation is performed during $(C \rightarrow D)$ and $(G \rightarrow H)$. The colour contours indicate the topology derivative and the wireframe denote the coarse resolution mesh of each new topology. The final optimised shapes are shown as smooth shaded surfaces.



(a) Initial domain (b) Deformed domain

Figure 6.19.: Interior free boundary value problem for the Laplace equation. The boundary of the domain Ω consists of a fixed part Γ_0 and a free part Γ_f. During optimisation, the original problem domain Ω_0 is deformed to $\Omega \to \Omega_t$ with a corresponding change to the free boundary $\Gamma_f \to \Gamma_{f,t}$.

6.5. BEM-based shape optimisation in electrostatics

In electrostatics, interior free boundary problems of Bernoulli type are given by the following overdetermined Laplace equation [66]

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma_0 \\ u = 1,\ \frac{\partial u}{\partial \boldsymbol{n}} &= Q \quad \text{on } \Gamma_f \end{aligned} \tag{6.28}$$

where u represents the electrostatic potential and $Q \geq 0$ is a prescribed constant expected value. The boundary of the domain Ω consists of a fixed part Γ_0 and a free part Γ_f as indicated in Figure 6.19a. The problem stated in (6.28) over constrained boundary conditions on Γ_f. In a physical interpretation, the shape of the free boundary Γ_f, with a prescribed electric potential, must be determined such that the electrostatic flux $\partial u/\partial \boldsymbol{n}$ along it is constant. A theoretical study of solutions to such interior problems can be found in [50]. In the related exterior problem, the free boundary Γ_f is located exterior(c.f. Figure 6.19a) to the fixed boundary Γ_o with prescribed potential. This class of problems can be reformulated as an optimisation problem [66, 75, 44] where the state equation is given by the Dirichlet boundary value problem for the Laplace equation

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma_0 \\ u &= 1 \quad \text{on } \Gamma_f \end{aligned} \tag{6.29}$$

with the objective function

$$\mathcal{J}(\boldsymbol{x}, u(\boldsymbol{x})) = \frac{1}{2} \left\| \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) - Q \right\|^2_{L^2(\Gamma_f)} = \frac{1}{2} \int_{\Gamma_f} \left(\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) - Q \right)^2 \mathrm{d}\,\Gamma_f \tag{6.30}$$

6.5.1. Shape derivative and computation

Using the notation introduced in Section 4.2, let $t\delta\boldsymbol{v}$ represent a design change in shape resulting in the domain being deformed from the initial configuration Ω_0 to the current configuration Ω_t. The perturbed problem is now restated using the deformed domain;

$$\begin{aligned} -\Delta u_t &= 0 \quad \text{in } \Omega_t \\ u_t &= 0 \quad \text{on } \Gamma_{0,t} \\ u_t &= 1 \quad \text{on } \Gamma_{f,t} \end{aligned} \tag{6.31}$$

with a corresponding change of the objective function (6.30) evaluated on the perturbed domain given by

$$\mathcal{J}(\boldsymbol{x}_t, u(\boldsymbol{x}_t)) = \frac{1}{2} \int_{\Gamma_{f,t}} \left(\frac{\partial u_t}{\partial \boldsymbol{n}_t}(\boldsymbol{x}_t) - Q \right)^2 \mathrm{d}\,\Gamma_{f,t} \tag{6.32}$$

The shape derivative (4.4) can be expressed as a boundary integral in an analogous manner to the linear elasticity shape derivative in Section 6.1

$$D\mathcal{J}(\boldsymbol{x}, u(\boldsymbol{x}))[\delta\boldsymbol{v}] = \int_{\Gamma_f} f(u, \lambda)(\delta\boldsymbol{v} \cdot \boldsymbol{n})\, \mathrm{d}\Gamma_f \tag{6.33}$$

The exact form of $f(u, \lambda)$ for the problem 6.31 and the objective function 6.30 under consideration is given by [5, 130, 45]

$$f(u, \lambda) = -\frac{\partial \lambda}{\partial \boldsymbol{n}}(\boldsymbol{x}) \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) - \frac{H(\boldsymbol{x})}{2} \left(\left(\frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}) \right)^2 - Q^2 \right) \tag{6.34}$$

In the above expression, H is the curvature and λ is the solution of the adjoint problem

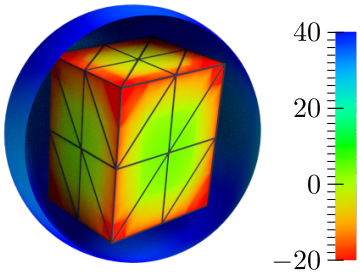
$$\begin{aligned} -\Delta\lambda &= 0 && \text{in } \Omega \\ \lambda &= 0 && \text{on } \Gamma_0 \\ \lambda &= \frac{\partial u}{\partial \boldsymbol{n}} - Q && \text{on } \Gamma_f \end{aligned} \tag{6.35}$$

A brief outline of boundary elements methods is presented in Appendix E.2, see Bandara et al. [5] for more details of the particular implementation used in the present work.

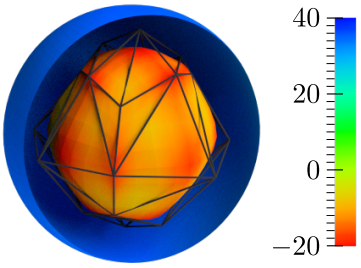
In the present work, a direct boundary element formulation based on Green's formula [5] is used for solving the direct (6.31) and adjoint (6.35) problems necessary for evaluating the shape derivative (6.33). In a regular BEM setting, the system matrices are fully populated restricting the problem size that can be efficiently computed. The curvature H required for the shape derivative (6.34) is estimated with discrete differential operators as given in Meyer et al. [93].

6.6. Electrostatic shape optimisation examples

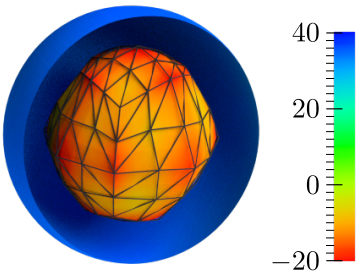
Two example problems are presented, the first is an introductory example with a known optimal shape and the other being a real-world example inspired by the Corporate Research Department of ABB Switzerland Ltd. located in Baden–Dättwil, Switzerland. During the optimisation only the shape of the boundary $\Gamma_{f,t}$ is updated with subdivision based multiresolution shape optimisation described in Section 4.3. Note that only Loop subdivision is used in the following examples. The boundary Γ_0 and its mesh resolution remains unchanged. As in previous cases, the discretised optimisation problem is solved with the MMA algorithm [124]. The input to the used nlopt optimisation library [76] consists of the cost function $J(\boldsymbol{x}^{\ell_c}, u(\boldsymbol{x}^{\ell_c}))$ evaluated in the fine resolution and the position vectors $\boldsymbol{x}_i^{\ell_o}$ and the gradients $\boldsymbol{g}_i^{\ell_o}$ for each coarse geometry node in the free boundary $\Gamma_f^{\ell_o}$. In addition suitable geometric bounds for the design variables are provided to the nlopt library.



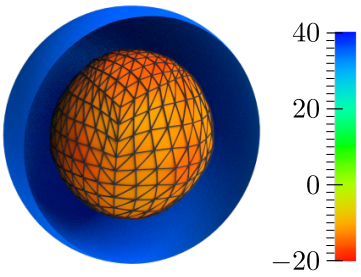
(a) Initial geometry at $\ell_o = 0$.



(b) Optimised geometry at $\ell_o = 0$.



(c) Optimised geometry at $\ell_o = 1$.



(d) Final optimised geometry at $\ell_o = 2$.

Figure 6.20.: Box in a sphere. Initial and optimised geometries with isocontours of the normal flux. The shown meshes indicate the optimisation level ℓ_o. The shown isocontours belong to the fine computational mesh at $\ell_c = 2$. The geometries shown in (b) and (c) represent intermediate results and (d) represents the final result.

6.6.1. Box in a sphere

This introductory example consists of optimising the shape of a box placed inside a sphere. It can be shown that the optimal shape for the box is a sphere. The box, representing $\Gamma_{f,t}$ has dimensions $0.16 \times 0.2 \times 0.24$ and the outer sphere, representing Γ_0, has radius 0.2. The coarse mesh for the box contains 48 elements which increases to 768 elements in the twice subdivided fine mesh at level $\ell_c = 2$. During the subdivision refinement the creases in the coarse mesh are maintained as creases using extended Loop subdivision stencils [14]. Note that on the limit surface the creases are only C^0-continuous. The resolution of the outer sphere remains fixed with 320 elements. Hence, the meshes for the boundary element analysis of the cube and sphere consist of 768 and 320 elements, respectively.

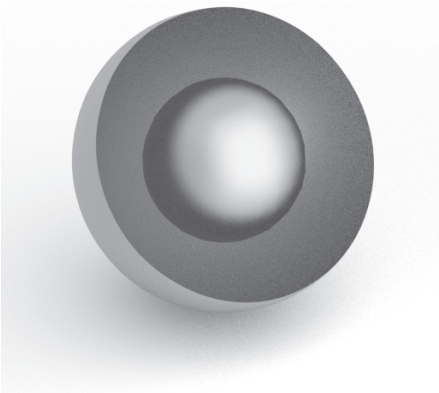
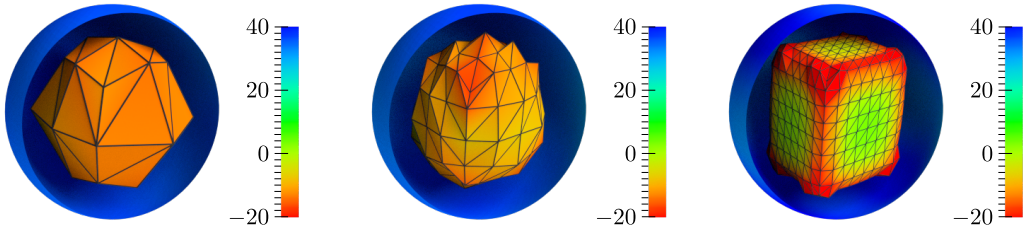


Figure 6.21.: Box in a sphere. Rendering of the limit surface of the final optimised shape.

Figure 6.20a shows the initial coarse geometry yielding a cost function value of 52.37. Note that the expected value Q in (6.30) is set to 20. First this coarse geometry is selected as optimisation level, i.e. $\ell_o = 0$, resulting in the optimised geometry shown in Figure 6.20b. After consecutively selecting $\ell_o = 1$ and $\ell_o = 2$ and optimising the final optimised geometry shown in Figure 6.20d is obtained. This final shape of the initial box is nearly a sphere of radius 0.215 and the cost function value is 16.48, which represents a reduction of 68.54%. As to be expected, the optimisation leads to a geometry with nearly uniform distribution of normal flux, see Figure 6.20d. In Figure 6.21 a rendering of the limit surface of the final optimised geometry is shown. Notice the high smoothness of the optimised geometry.

In order to demonstrate the robustness and benefits of the proposed multiresolution optimisation approach the problem is recomputed using the same level for optimisation and computation, i.e. $\ell_o \equiv \ell_c$. Figure 6.22 shows the geometries obtained with $\ell_o \equiv \ell_c \equiv 0$, $\ell_o \equiv \ell_c \equiv 1$ and $\ell_o \equiv \ell_c \equiv 2$. The corresponding reduction of the objective function are 45.2%, 67.41%, 31.34%, respectively. Notice in Figure 6.22 the stark differences in the three geometries obtained and the unphysical geometry oscillations for the finer meshes. Moreover, the result obtained with $\ell_o \equiv \ell_c \equiv 0$ is highly questionable because of the coarseness of the computational mesh. The unphysical oscillations obtained in Figure 6.22b are reminiscent of problems reported in the finite element context [18, 63].



(a) $\ell_o \equiv \ell_c \equiv 0$ (b) $\ell_o \equiv \ell_c \equiv 1$ (c) $\ell_o \equiv \ell_c \equiv 2$

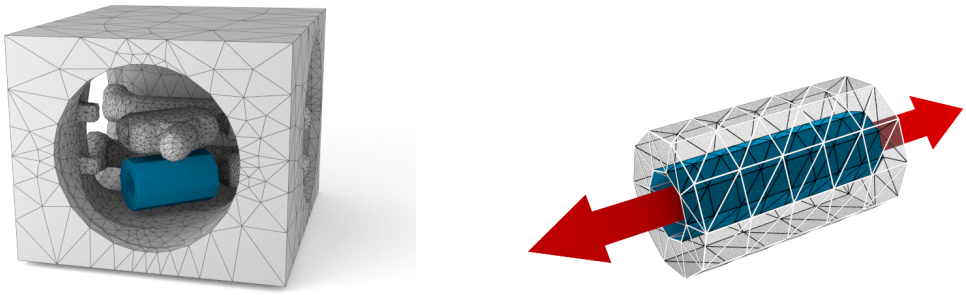
Figure 6.22.: Box in a sphere. Optimisation without multiresolution using the same level for shape control and computation, i.e. $\ell_o \equiv \ell_c$. The shown meshes indicate the level $\ell_o \equiv \ell_c$. Isocontours indicate the normal flux.

6.6.2. Gas insulated switchgear

In this example, the proposed multiresolution shape optimisation framework is applied to the design of an electrode in an gas insulated switchgear component, Figure 6.23a. Such electromechanical components are widely used as circuit breakers in high-voltage power transmission. The objective of shape optimisation is to reduce the propensity of the component for electric breakdown with the ultimate aim to enable more compact component geometries. This can be achieved by modifying the electrode geometries so that the maximum normal flux is minimised. This is substituted by minimising the L_2-norm of the normal flux (6.30) as previously explained.

In Figure 6.23 the gas insulated switchgear component with the electrode in the form of a primitive cylinder is shown. The cylinder represents the electrode geometry $\Gamma_{f,t}$ to be optimised. The initial coarse mesh of the cylinder contains 264 elements. The creases on the cylinder are not tagged so that the geometry becomes smoother while it is subdivision refined. As a design constraint, the inner surface of the cylinder is required to have a constant radius for a bolt passing through it. Geometric bounds on the positions of vertices lying on the inner surface are applied to prevent any radial movement that would violate this design requirement, Figure 6.23b.

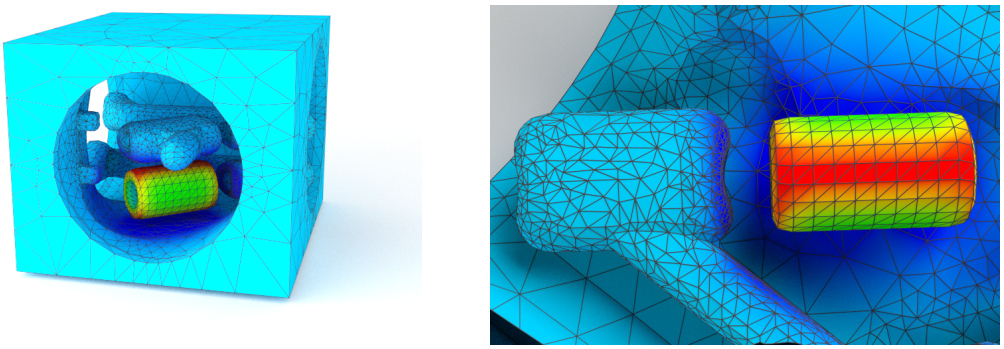
The once subdivision refined mesh with 1056 elements is chosen as the computational level, i.e. $\ell_c = 1$. As can be seen in Figure 6.24, the ends of the cylinder become smoother because the usual (vs. extended) subdivision stencils are applied throughout the mesh. In this example, we consider the geometry at level $\ell_o = 0$ for optimisation. In the initial design, Figure 6.24, the maximum normal flux is $J_{\max} = 81.63$ before optimisation, and



(a) The cylindrical electrode boundary $\Gamma_{f,t}$ to be optimised is shown in dark blue and the other surfaces representing Γ_0 are shown in light grey.

(b) Close-up of the cylindrical electrode. Vertices on the inner surface of the cylinder shown in dark blue are only allowed to move along the axis of the cylinder.

Figure 6.23.: Gas insulated switchgear. Initial geometry and geometric optimisation constraints.

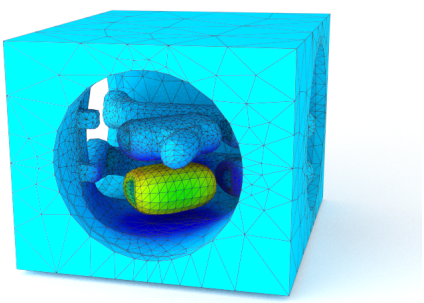


(a) Overall component

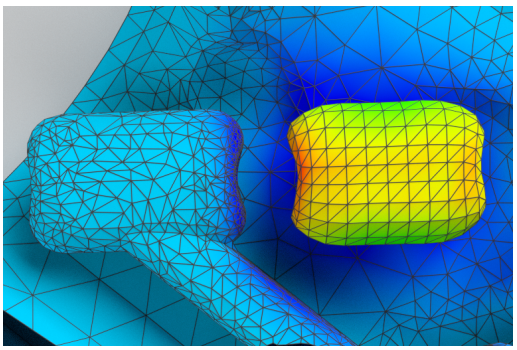
(b) Close-up of the electrode.

Figure 6.24.: Gas insulated switchgear. Isocontours of the normal flux for the initial cylindrical electrode design.

reduces to $J_{\max} = 66.99$ in the optimised shape shown in Figure 6.25, corresponding to a reduction of 17.94%. However the reduction of the objective function L_2-norm of the normal flux is much higher being 38.24%. Figure 6.26 shows the component with the electrode geometry as currently manufactured by ABB. This electrode geometry has been obtained over the years by combining engineering intuition with simple calculations and testing. The similarities between the methodically shape optimised and the electrode geometry in production are striking . Notice in particular the saddle shape at the two ends of the original cylinder which helps to lower the large normal flux at the sharp crease at the boundary of the inner hole.

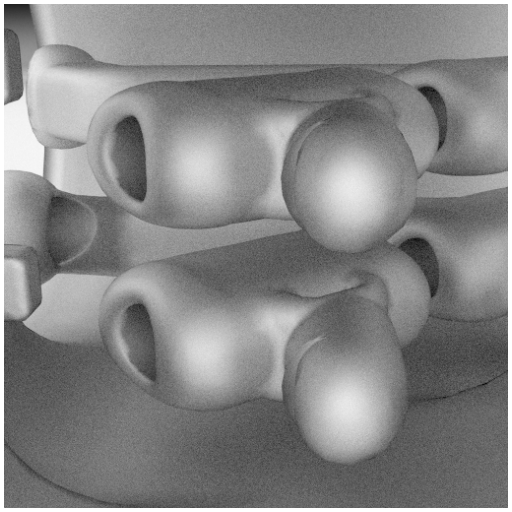


(a) Overall component

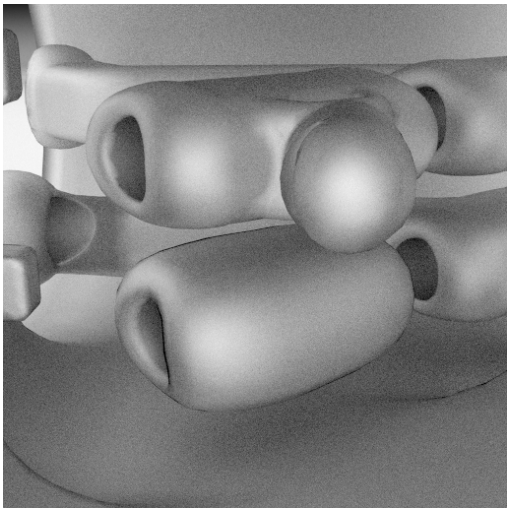


(b) Close-up of the electrode.

Figure 6.25.: Gas insulated switchgear. Isocontours of normal flux for the optimised electrode design.



(a) Manually optimised electrode.



(b) Limit surface of the multiresolution optimised electrode.

Figure 6.26.: Gas insulated switchgear. Comparison of multiresolution shape optimised design with manually optimised electrode produced by ABB.

7. Conclusions and Future Research

7.1. Conclusions

This dissertation proposes the use of a hierarchy of meshes with varying resolutions for integrated geometric design, analysis and optimisation of shells and solids. The motivation behind the multiresolution framework were twofold: (*i*) facilitate the seamless integration of different software modules required during the iterative product development, and (*ii*) avoid the one-to-one correspondence between the geometric design variables and the finite element mesh to facilitate parameter-free shape optimisation of shells and solids. Functionality of the proposed multiresolution framework was demonstrated for a variety of validation and industry strength examples in shell and solid optimisation. In summary, the following conclusions can be made:

- In practice, changes to a geometric design need to be made iteratively based on simulation and optimisation results. A multiresolution framework based on subdivision surfaces (Section 4.3) has been developed in this dissertation as a suitable solution for such scenarios. The subdivision refinement operation provides a natural means for propagating information from coarse to fine resolutions. The reverse subdivision operation can be performed with least squares fitting as used in the present work (Section 4.3.3). The introduced subdivision surface based framework requires the starting geometry to be a coarse resolution mesh. Hence this is essentially a top-down approach where the multiresolution hierarchy is constructed starting from a coarse resolution, down to a fine resolution.

- This dissertation has also introduced an alternative bottom-up approach (Section 4.4) where the initial input is a fine resolution mesh of arbitrary connectivity containing many small-scale geometric features. This is desirable in situations

where various geometric design features are already present by the time the optimisation stage is reached. In this setting, the multiresolution hierarchy is created in a bottom-up manner starting from the given fine resolution mesh. A suitable framework was assembled to this end by combining quadric based mesh decimation with Laplacian smoothing following the progressive mesh approach (Chapter 3).

- In both top-down and bottom-up approaches introduced, the position vectors of a coarse resolution mesh are used as the design variables while analysis is performed using a fine resolution. The optimisation process is controlled by this difference in resolutions and the corresponding subdivision (2.14) or smoothing (3.10) operations used in translating data from coarse to fine resolutions. This enables parameter-free optimisation without any additional mesh or geometry smoothing typically required to obtain meaningful results from shape optimisation (c.f. Figure 6.22).

- Shape optimisation is required during various stages of product development. Different stages require varying degrees of emphasis on globalisation of geometric design features. In initial design stages, large shape changes are expected during optimisation whereas more minimal and localised shape changes are expected during later design stages. In the presented examples, this is demonstrated by gradually diminishing the difference between coarse and fine resolutions. Essentially the design variables are migrated to progressively finer resolutions during optimisation. This also eliminates mesh dependency of the optimised result to some extent (see Figures 5.6 and 6.2).

- Parameter-free shape optimisation of Kirchhoff-Love shells is demonstrated using the proposed multiresolution frameworks in Chapter 5. For the shells, analytical shape derivatives are derived (Section 5.3) with respect to the position vector of each node in the finite element mesh. A subdivision based isogeometric solver module is used for solving the mechanical problem and computing the shape derivatives. Shape optimisation of a real world example (Section 5.4.3) is presented where various alternative design constraints are present. It is shown how these design constraints can be incorporated in each of the multiresolution frameworks underlining their potential in industrial applications.

- Use of standard finite elements for computing solid objects require a volumetric mesh which is likely to be distorted during shape optimisation. Solid optimisation

is presented in this dissertation by using either immersed or boundary element methods (Chapter 6). A volumetric mesh is not required and a multiresolution hierarchy is created to represent only the boundary of the solid domain. Various shape (Section 6.2) and topology (Section 6.3) optimisation examples of linear elastic solids are presented to demonstrate the robustness of combining the multiresolution optimisation approach with immersed methods.

7.2. Future Research

Several extensions and improvements of the proposed multiresolution optimisation approach are listed below:

- The introduced top-down subdivision based optimisation approach is only capable of accepting an initial coarse resolution subdivision control mesh. It is possible to extend this to include initial fine resolution meshes. The necessary input mesh can be created by fitting a mesh with subdivision connectivity to the original input mesh [43, 87]. The difference between the original and the fitted mesh can be expressed in detail vectors (4.9) and stored in local frames as demonstrated in Section 3.3.2. A similar method is described in Zorin et al. [132].

- A limitation of the current bottom-up framework is that coarse resolution field data is only a subset of the fine resolution data without any averaging or smoothing (Section 4.4.3). This can be addressed as suggested in Figure 4.14 by adding a pre-smoothing step. However having two smoothing steps may result in longer details and amplification of the non-uniform geometry editing effects highlighted in Section 4.4.2.

- In addition to the subdivision and progressive mesh multiresolution frameworks presented, alternative schemes can be developed. NURBS compatible subdivision [20] offers an attractive option in this context as it combines features of both NURBS and subdivision.

- Finally, the extension to topology optimisation presented in Section 6.4 requires manual generation of the new topology boundary. This process can be automated

by extracting iso-contours of the topology derivative (6.26). This implies the extracted surface has approximately the same element size as the background mesh requiring use of the progressive mesh approach in subsequent optimisation steps.

A. Appendix

A.1. Eigenanalysis of subdivision matrix

The study of the subdivision surface geometry near the vicinity of a particular vertex requires knowing only the control points in its local neighbourhood. Specifically the one-ring neighbours, all vertices sharing an edge with the vertex in question, are required in Loop and Catmull-Clark subdivision. Let $\mathbf{x}^0$ be coordinates of all the control points of such a neighbourhood. The effect of subdividing $\mathbf{x}^0$ infinite times results in the control polygon converging to the surface

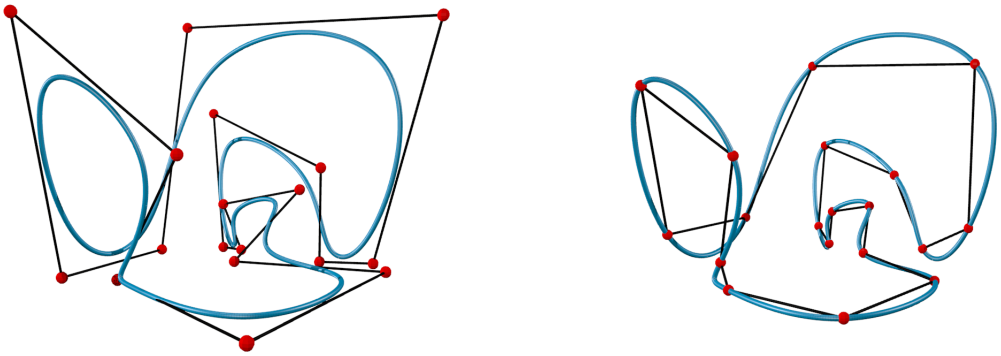
$$\mathbf{x}^\infty = \lim_{n\to\infty} \mathbf{S}^n \mathbf{x}^0 \tag{A.1}$$

The properties of the subdivision matrix can be analysed via an eigen-decomposition. Let $\boldsymbol{\lambda_I} = \{\lambda_0, \lambda_1, \ldots, \lambda_{n-1}\}$, $\boldsymbol{L_I} = \{\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_{n-1}\}$ and $\boldsymbol{R_I} = \{\boldsymbol{R}_0, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_{n-1}\}$ be the eigenvalues and the left and right eigenvectors of $\mathbf{S}$, respectively with $\lambda_i \geq \lambda_{i+1}$. The control polygon $\mathbf{x}^0$ can be expressed as linear combination of eigenvectors

$$\mathbf{x}^0 = \sum_{i=0}^{n-1} a_i^0 \boldsymbol{R}_i, \qquad a_i^0 = \boldsymbol{L}_i \cdot \boldsymbol{x}^0 \tag{A.2}$$

Now (A.1) can be expressed as

$$\begin{aligned}
\mathbf{x}^\infty &= \lim_{n\to\infty} \mathbf{S}^n \sum_{i=0}^{n-1} a_i^0 \boldsymbol{R}_i \\
&= \lim_{n\to\infty} \sum_{i=0}^{n-1} a_i^0 \boldsymbol{S}^n \boldsymbol{R}_i \\
&= \lim_{n\to\infty} \sum_{i=0}^{n-1} a_i^0 \lambda_i^n \boldsymbol{R}_i
\end{aligned}$$



(a) Control points and limit curve.

(b) Control points projected onto the limit curve.

Figure A.1.: Application of 1D cubic limit masks. Control points of non-interpolating subdivision schemes like Loop, Catmull-Clark can be projected in the limit curve/surface using limit masks.

The above eigenanalysis reveals several interesting properties about subdivision:

Convergence: For the subdivision scheme to be convergent, the largest eigenvalue $\lambda_0 > \lambda_i$ has to be $\lambda_0 = 1$.

Affine invariance: $\lambda_0 = 1$ and $\boldsymbol{R}_i = \{1, 1, \ldots, 1\}$ for affine invariance, i.e. affine transformation is applied to the subdivision surface by applying it to the control points.

Limit position: The limit position of $\boldsymbol{x}^0$ is

$$\mathbf{x}^\infty = a_i^0 \boldsymbol{R}_i = a_i^0 = \boldsymbol{L}_0 \cdot \mathbf{x}^0 \tag{A.3}$$

The affine combinations in $\boldsymbol{L}_0$ can be used in a similar way to subdivision masks and can be expressed using *limit masks*. This results in the control points being projected in the limit curve/surface, see Figure A.1.

Tangents: The tangents at the limit position are

$$\boldsymbol{t}_1 = \boldsymbol{L}_1 \cdot \mathbf{x}^0, \qquad \boldsymbol{t}_2 = \boldsymbol{L}_2 \cdot \mathbf{x}^0 \tag{A.4}$$

and the surface normal is

$$\boldsymbol{n} = \boldsymbol{t}_1 \times \boldsymbol{t}_2 \tag{A.5}$$

A more detailed discussion of the above can be found in [131].

B. Appendix

B.1. Incremental mesh decimation with quadrics

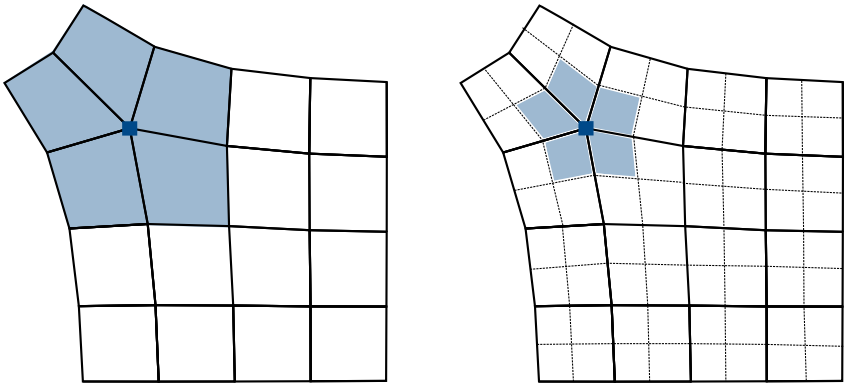
The following describes an algorithm for incremental decimation using halfedge collapse. It is assumed some binary oracle is present and quadric based error measures are used as the continuous oracle. The original algorithm was presented by Garland [54] for general edge collapse.

1. Allocate initial quadric $\mathbf{Q}_i^n$ (3.7) for each vertex $\boldsymbol{x}_i^n$ in starting mesh Ω^n.
2. Collect list of all halfedge collapses.
3. Use binary oracles to eliminate disallowed halfedge collapses.
4. Compute error for each halfedge collapse $(\boldsymbol{x}_i^n, \boldsymbol{x}_j^n) \rightarrow \boldsymbol{x}_i^{n-1}$ in list:
   - The error of removing $\boldsymbol{x}_j^n$ during the collapse can be computed using the sum of quadrics of the two vertices $(\hat{\boldsymbol{x}}_j^n)^T(\mathbf{Q}_i^n + \mathbf{Q}_j^n)\hat{\boldsymbol{x}}_j^n$.
5. Form priority queue:
   - The list of allowable halfedge collapses is sorted by ascending error.
6. Collapse halfedge at the top of the priority queue:
   - After the halfedge $(\boldsymbol{x}_p^n, \boldsymbol{x}_q^n) \rightarrow \boldsymbol{x}_p^{n-1}$ is collapsed, the quadric of vertex $\boldsymbol{x}_p^{n-1}$ is updated; $(\mathbf{Q}_p^n + \mathbf{Q}_q^n) \rightarrow \mathbf{Q}_p^{n-1}$.
7. Go to step 2.

Note that the binary oracle needs to be re-evaluated only for halfedges in the neighbourhood of the previous collapse due to local geometry changes. For example, the aspect ratio of elements will be different and will change the outcome of a binary oracle for aspect ratio.

C. Appendix

C.1. Exact evaluation of subdivision surfaces

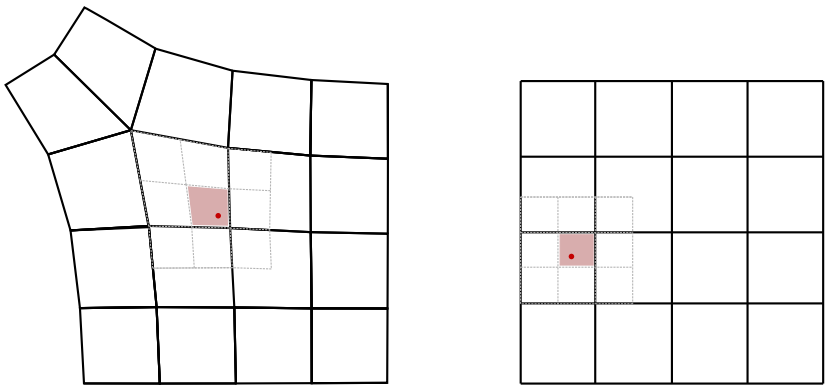


(a) Irregular region before subdivision.

(b) Irregular region after subdivision.

Figure C.1.: A Catmull-Clark control mesh after a single subdivision. Initially, the five elements connected to the extraordinary vertex (blue square) are irregular and cannot be evaluated as a bi-cubic B-spline patch. However after a single subdivision step, the area covered by irregular elements (blue) is reduced.

Catmull-Clark subdivision is equivalent to uniform B-splines in the regular setting and only needs special rules for exact evaluation in the vicinity of extraordinary vertices. An efficient method for exact evaluation in such regions was proposed by Stam [121]. Subdivision is equivalent to midpoint uniform B-spline knot insertion, implying that after sufficient number of subdivisions any point in the irregular region can be made to coincide with a known parametric representation. For example in Catmull-Clark surfaces, it is a bi-cubic B-spline patch.



(a) Near extraordinary Vertex (b) Near boundary

Figure C.2.: Exact evaluation of Catmull-Clark subdivision. The mesh is subdivided (only once here) until evaluation point (red circle) is within a regular element. Any point within the regular element can be evaluated using a bi-cubic B-spline patch (grey).

Let $\mathbf{x}^o$ denote coordinates of the patch of control vertices in the irregular region. A single subdivision is equivalent to multiplication by subdivision matrix $\mathbf{S}$

$$\mathbf{x}^1 = \mathbf{S}\mathbf{x}^0 \tag{C.1}$$

After a single subdivision step, 3 of the 4 new elements in the subdivided irregular element are regular as shown in Figure C.1. This can be repeated (say ℓ times) until the required evaluation point belongs to a regular vertex patch (Figure C.2), which can be extracted using a picking matrix $\mathbf{P}$ comprising of 1's and 0's.

$$\hat{\mathbf{x}}^{\ell} = \mathbf{P}\mathbf{S}^{\ell}\mathbf{x}^0 = \mathbf{P}\mathbf{S}^{\ell}\mathbf{x}^0 \tag{C.2}$$

where $\hat{\mathbf{x}}^{\ell}$ contains the vertices corresponding to the known B-spline patch with the shape function vector $\boldsymbol{\Phi}$. The desired evaluation point can now be obtained using

$$\boldsymbol{x}(\boldsymbol{\theta_1}, \boldsymbol{\theta_2}) = \boldsymbol{\Phi}^{\mathsf{T}}(\tilde{\boldsymbol{\theta}}_1, \tilde{\boldsymbol{\theta}}_2)\hat{\mathbf{x}}^{\ell} = \boldsymbol{\Phi}^{\mathsf{T}}(\tilde{\boldsymbol{\theta}}_1, \tilde{\boldsymbol{\theta}}_2)\mathbf{A}^{\ell}\mathbf{x}^0 \tag{C.3}$$

The compact supports of B-splines imply that the interpolation of a given element does not depend on the entire mesh. Only a particular neighbourhood of the element, for example the one ring neighbours for Loop and Catmull-Clark, is sufficient. When using (C.3), $\mathbf{x}^0$ is replaced by this local neighbourhood and yields the required regular

patch $\hat{\mathbf{x}}^\ell$ used for parametrisation using a known B-spline patch $\boldsymbol{\Phi}$

$$\boldsymbol{x_h(\theta_1, \theta_2)} = \sum_{i=1}^{k} N_i(\theta_1, \theta_2)\boldsymbol{x}_i, \quad \boldsymbol{x}_i \in \mathbf{x}^0 \tag{C.4}$$

where k is the size of the local neighbourhood. The subdivision shape functions are given by $N_i = (\boldsymbol{\Phi}^\mathsf{T}\mathbf{A}^\ell)_i$. The derivatives can be computed as follows

$$\frac{\partial \mathbf{N}_i(\theta_1, \theta_2)}{\partial \theta_j} = \frac{\partial (\boldsymbol{\Phi}^\mathsf{T}\mathbf{A}^\ell)_i}{\partial \theta_j} = \left(\frac{\boldsymbol{\Phi}(\tilde{\theta}_1, \tilde{\theta}_2)}{\partial \tilde{\theta}_i} \frac{\partial \tilde{\theta}_i}{\partial \theta_j} \mathbf{A}^\ell \right)_i, \quad j \in \{1, 2\} \tag{C.5}$$

Computation of the subdivision matrix

The exact evaluation described above requires the subdivision matrix $\mathbf{S}$ and picking matrix $\mathbf{P}$ for which either of the following two methods can be employed;

- *Maintain a particular ordering for the vertices in a patch*
  This implies the subdivision matrix has a particular block structure and can be evaluated directly through eigen analysis similar to the approach by Stam [121] for Catmull-Clark surfaces. The picking matrix $\mathbf{P}$ can be explicitly defined to *pick* the regular vertex patch for the requred element. However this process is very complicated for extended subdivison surfaces with corner, crease and edge tags.

- *Compute* $\mathbf{P}$ *and* $\mathbf{S}^n$
  Examining (C.1), it is obvious that each column of $\mathbf{S}$ can be computed by setting a value of 1 for one vertex and 0 in others [31]. Let $\mathbf{x}^0$ denote the starting mesh with n vertices. Set a unit value for the J^{th} vertex

  $$\mathbf{x}^0 = (\mathbf{0}, \mathbf{0}, \ldots, \boldsymbol{x}_J, \ldots, \mathbf{0}), \quad \boldsymbol{x}_J = \mathbf{1},\ J \in [1, n] \tag{C.6}$$

  and subdivide once; $\mathbf{x}^1 = \mathbf{S}\mathbf{x}^0$. Now the J^{th} column of the subdivision matrix $\boldsymbol{S}_J$ is given by

  $$\boldsymbol{S}_J = \boldsymbol{x}^1 \tag{C.7}$$

  Similarly, $\mathbf{A}^\ell$ can also be computed directly using the previous method. Simply collect the values of the required regular patch after ℓ subdivisions of the initial

mesh with of 1's and 0's set as in (C.6);

$$\mathbf{A}_J^\ell = \hat{\mathbf{x}}^\ell \tag{C.8}$$

D. Appendix

D.1. Shape derivatives of covariant and contravariant basis vectors

The covariant basis (5.3) is defined using the finite element discretisation (5.19)

$$\boldsymbol{a}_\alpha = \sum_{i=1}^{k} N_{i,\alpha} \boldsymbol{x}_i \tag{D.1}$$

The partial derivative of the covariant basis with respect to a design variable $\boldsymbol{x}_i$ is given by

$$\boldsymbol{a}_{\alpha,\boldsymbol{x}_i} = N_{i,\alpha} \mathbf{1} \tag{D.2}$$

Note that in practice, the derivatives with respect to each component in $\boldsymbol{x}_i$ are required, this is not distinguished in the present work for brevity. The derivative of the unit normal is more involved;

$$\begin{aligned} \boldsymbol{a}_{\boldsymbol{3},\boldsymbol{x}_i} &= \frac{\partial \left[\frac{\boldsymbol{a_1} \times \boldsymbol{a_2}}{|\boldsymbol{a_1} \times \boldsymbol{a_2}|} \right]}{\partial \boldsymbol{x}_i} \\ &= \frac{(\boldsymbol{a_1} \times \boldsymbol{a_2})_{,\boldsymbol{x}_i} |\boldsymbol{a_1} \times \boldsymbol{a_2}| - (\boldsymbol{a_1} \times \boldsymbol{a_2}) |\boldsymbol{a_1} \times \boldsymbol{a_2}|_{,\boldsymbol{x}_i}}{|\boldsymbol{a_1} \times \boldsymbol{a_2}|^2} \end{aligned} \tag{D.3}$$

with the derivative of the magnitude

$$|\boldsymbol{a_1} \times \boldsymbol{a_2}|_{,\boldsymbol{x}_i} = \frac{(\boldsymbol{a_1} \times \boldsymbol{a_2}) \cdot (\boldsymbol{a_1} \times \boldsymbol{a_2})_{,\boldsymbol{x}_i}}{|\boldsymbol{a_1} \times \boldsymbol{a_2}|} \tag{D.4}$$

Finally the derivative of the cross product is

$$(\boldsymbol{a_1} \times \boldsymbol{a_2})_{,\boldsymbol{x}_i} = (\boldsymbol{a}_{\boldsymbol{1},\boldsymbol{x}_i} \times \boldsymbol{a_2}) + (\boldsymbol{a_1} \times \boldsymbol{a}_{\boldsymbol{2},\boldsymbol{x}_i}) \tag{D.5}$$

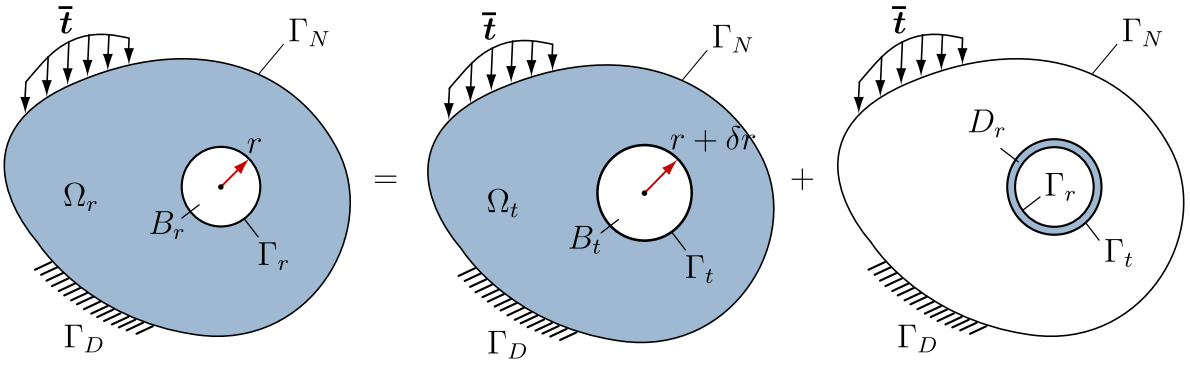


Figure D.1.: Domain truncation

The derivative of the contravariant basis requires the definition of the Jacobian of the mapping $\mathbf{J} = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \boldsymbol{a}_3]^\mathsf{T}$. From the definition of the contravariant basis $\bar{\boldsymbol{a}}^i \cdot \bar{\boldsymbol{a}}_j = \delta^i_j$, it follows that

$$[\boldsymbol{a}^1\ \boldsymbol{a}^2\ \boldsymbol{a}^3]^\mathsf{T}_{,\boldsymbol{x}_i} = (\mathbf{J}^\mathsf{T})^{-1}_{,\boldsymbol{x}_i} \tag{D.6}$$

For non-singular $\mathbf{J}$, the following relationship holds;

$$\begin{aligned} \mathbf{J}^\mathsf{T}(\mathbf{J}^\mathsf{T})^{-1} &= \mathbf{I} \\ \mathbf{J}^\mathsf{T}_{,\mathbf{x}_i}(\mathbf{J}^\mathsf{T})^{-1} + \mathbf{J}^\mathsf{T}(\mathbf{J}^\mathsf{T})^{-1}_{,\mathbf{x}_i} &= 0 \end{aligned} \tag{D.7}$$

The derivative of the inverse of the Jacobian can now be computed from;

$$(\mathbf{J}^\mathsf{T})^{-1}_{,\mathbf{x}_i} = -(\mathbf{J}^\mathsf{T})^{-1}\mathbf{J}^\mathsf{T}_{,\mathbf{x}_i}(\mathbf{J}^\mathsf{T})^{-1} \tag{D.8}$$

D.2. Topological-shape sensitivity method

Introduced by Novotny et al. [99], the topological-shape sensitivity method represents the topology derivative using the shape sensitivity. The key idea is that the hole B_r already exists and is perturbed by a small amount δr, creating a new hole B_t and a domain $\Omega_t = \Omega \setminus B_t$. It is evident that $\Omega_t|_{t=0} = \Omega_r$. The following transformation can be used to represent this mapping between Ω_t and Ω_r (c.f. Section 4.2)

$$\boldsymbol{x}_t = \boldsymbol{x} + t\delta\boldsymbol{v}, \quad \boldsymbol{x}_t \in \Gamma_t,\ \boldsymbol{x} \in \Gamma_r \tag{D.9}$$

The shape change in this context is restricted to the expansion of the hole B_τ, giving the following definition for design velocity $\delta\boldsymbol{v}$

$$\delta\boldsymbol{v} = -\boldsymbol{n} \quad \text{on } \Gamma_r \tag{D.10}$$

Shape sensitivity of the objective function $\mathcal{J}(\Omega_r)$ with respect to the perturbation $t\delta\boldsymbol{v}$ reads

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}t}(\Omega_t)\bigg|_{t=0} = \lim_{t\to 0}\frac{\mathcal{J}(\Omega_t) - \mathcal{J}(\Omega_r)}{t} = \frac{\mathrm{d}\mathcal{J}}{\mathrm{d}r}(\Omega_r) \tag{D.11}$$

Note that the above is valid only for small t. The relationship between the topology derivative (6.26) and the above shape sensitivity of the hole can be established by taking the derivative of 6.25a

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}r}(\Omega_r) = +f(r)'D_T(\boldsymbol{x_0}) + \mathcal{O}(f(r))'f'(r) \tag{D.12}$$

substituting (D.11) and rearranging

$$\begin{aligned}\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}t}(\Omega_t)\bigg|_{t=0} &= f'(r)\Big(D_T(\boldsymbol{x_0}) + \mathcal{O}(f(r))'\Big)\\ D_T(\boldsymbol{x}_0) &= \lim_{r\to 0}\frac{1}{f'(r)}\frac{\mathrm{d}\mathcal{J}(\Omega_t)}{\mathrm{d}t}\bigg|_{t=0}\end{aligned} \tag{D.13}$$

Refer to Novotny et al.[99, 101] for complete proof. A link between the topology derivative and shape sensitivity concepts is established in (D.13). The next task is to determine $f(r)$ such that $0 < |D_T(\boldsymbol{x}_0)| < \infty$ as $r \to 0$. Considering the original linear elastic boundary value problem (6.1) formulated in the truncated domain $D_r = \Omega_t \setminus \Omega_r$ (Figure D.1);

$$\begin{aligned}-\nabla\cdot\boldsymbol{\sigma}(\hat{\boldsymbol{u}}_r) &= 0 \qquad \text{in } D_r\\ \hat{\boldsymbol{u}}_r &= \psi \qquad \text{on } \Gamma_t\\ \boldsymbol{\sigma}(\hat{\boldsymbol{u}}_r)\cdot\boldsymbol{n} &= 0 \qquad \text{on } \Gamma_r\end{aligned} \tag{D.14}$$

with $\psi = \boldsymbol{u}_r|_{\Gamma_t}$, where $\boldsymbol{u}_r$ is the solution to the problem formulated in Ω_r (6.24). Note that the domain truncation concept from Garreau et al. [57] implies the following identity;

$$\boldsymbol{u}_r = \hat{\boldsymbol{u}}_r|_{D_r} \tag{D.15}$$

Essentially problems 6.24 and D.14 now have the same solution in D_r. The latter is given by the analytical solution for stress distribution around a circular/spherical void.

Given property D.15, an asymptotic analysis of (D.13) using the solution of D.14 as $r \to 0$ will give $f(r)$ and the topology derivative.

E. Appendix

E.1. Features of immersed method

The following is a brief outline of the immersed method [109, 110] used in Sections 6.2 and 6.4 as the solver module in shape and topology optimisation of linear elastic solids.

- The boundary Γ of the physical domain Ω is represented in the background grid ω using an implicit signed distance function ϕ

$$\phi(\boldsymbol{x}, \Gamma) = \begin{cases} \text{distance}(\boldsymbol{x}, \Gamma) & \text{if } \boldsymbol{x} \in \Omega \\ 0 & \text{if } \boldsymbol{x} \in \Gamma \\ -\text{distance}(\boldsymbol{x}, \Gamma) & \text{otherwise} \end{cases} \tag{E.1}$$

- In immersed methods, auxiliary procedures are required to integrate the physical domain boundaries into the solution from the block-structured background mesh. Consider the weak form of the linear elastic boundary value problem (6.1)

$$\int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \, \mathrm{d}\Omega = \int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}\Omega + \int_{\Gamma_N} \bar{\boldsymbol{t}} \cdot \boldsymbol{v} \, \mathrm{d}\Gamma + \int_{\Gamma_D} \boldsymbol{t}(\boldsymbol{u}) \cdot \boldsymbol{v} \, \mathrm{d}\Gamma \,. \tag{E.2}$$

  In classical finite element analysis, the last term disappears due to the assumption that $\boldsymbol{v} = \boldsymbol{0}$ on Γ_D. In [109], an integral equation formulation based on Nitsche's method is used to enforce essential boundary conditions.

- Tensor product B-splines of polynomial degree p are used as the shape functions on the background grid;

$$N_{\boldsymbol{i},p} = N_{i^1,p} \times \cdots \times N_{i^d,p} \tag{E.3}$$

where d is the dimension, $\boldsymbol{i} = \{i^1, \ldots, i^d\}$ is the multi-index of a grid point and $N_{i^1,p}, \ldots, N_{i^d,p}$ are the univariate B-splines (2.2).

- Let Ω_k denote a cut-cell which is partially inside the domain Ω, i.e. $\Omega_k \cap \Gamma \neq \emptyset$. As $|\Omega_k \cap \Gamma| \to 0$ the condition number of the system matrix increases which is addressed using a subdivision based stabilisation method method.

E.2. Introduction to boundary element methods

A boundary element solver using the fast multipole method presented in Of et al. [102] is used in Section 6.6 as the solver module. The boundary element formulation is based on the direct method using Green's formula, a brief outline of which is presented below following Hsiao and Wendland [72] and Banerjee [7].

Let $\omega(\boldsymbol{x}, \boldsymbol{y})$ denote the fundamental solution to the Laplacian in 3D given by

$$\omega(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{4\pi|\boldsymbol{x} - \boldsymbol{y}|} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3 \tag{E.4}$$

The solution of the boundary value problem (6.31) at any point inside the domain $u(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega$ is now given by the *boundary integral equation*

$$u(\boldsymbol{x}) = \int_\Gamma \omega(\boldsymbol{x}, \boldsymbol{y}) \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{y})\ \mathrm{d}\,\Gamma - \int_\Gamma \boldsymbol{u}(\boldsymbol{y}) \frac{\partial \omega}{\partial \boldsymbol{n}}(\boldsymbol{x}, \boldsymbol{y})\ \mathrm{d}\,\Gamma \quad \boldsymbol{x} \in \Omega,\ \boldsymbol{y} \in \Gamma \tag{E.5}$$

Similarly the solution on the boundary $u(\boldsymbol{x}),\ \boldsymbol{x} \in \Gamma$ takes the following form

$$\frac{1}{2} u(\boldsymbol{x}) = \int_\Gamma \omega(\boldsymbol{x}, \boldsymbol{y}) \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{y})\ \mathrm{d}\,\Gamma - \int_\Gamma \boldsymbol{u}(\boldsymbol{y}) \frac{\partial \omega}{\partial \boldsymbol{n}}(\boldsymbol{x}, \boldsymbol{y})\ \mathrm{d}\,\Gamma \quad \boldsymbol{x}, \boldsymbol{y} \in \Gamma \tag{E.6}$$

In contrast to the finite element method that contain domain integrals requiring Ω to be discretised, (E.5) and (E.6) contain only boundary integrals requiring a discretisation of only the boundary Γ. Assume the boundary is discretised into n_e elements, E.6 can be written for the k^{th} element as follows

$$\frac{1}{2} u(\boldsymbol{x}_k) = \sum_{i=1}^{n_e} \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{y}_i) \int_{\Gamma_i} \omega(\boldsymbol{x}_k, \boldsymbol{y}_i)\ \mathrm{d}\,\Gamma - \sum_{i=1}^{n_e} u(\boldsymbol{y}_i) \int_{\Gamma_i} \frac{\partial \omega}{\partial \boldsymbol{n}}(\boldsymbol{x}_k, \boldsymbol{y}_i)\ \mathrm{d}\,\Gamma \tag{E.7}$$

This can be expressed in matrix form and rearranged to yield a format similar to the standard discretised equilibrium equation given in (5.23)

$$\begin{aligned} \frac{1}{2}\mathbf{M}_h\mathbf{u}_h &= \mathbf{V}_h\mathbf{q}_h - \mathbf{K}_h\mathbf{u}_h \\ \mathbf{V}_h\mathbf{q}_h &= \left(\frac{1}{2}\mathbf{M}_h + \mathbf{K}_h\right)\mathbf{u}_h \end{aligned} \tag{E.8}$$

with

$$\mathbf{V}_h[i,j] = \int_{\Gamma_i}\int_{\Gamma_j} \omega(\boldsymbol{x}_i, \boldsymbol{y}_j)\,\mathrm{d}\,\Gamma_j\,\mathrm{d}\,\Gamma_i, \qquad \mathbf{K}_h[i,j] = \int_{\Gamma_i}\int_{\Gamma_j} \frac{\partial\omega}{\partial\boldsymbol{n}}(\boldsymbol{x}_i, \boldsymbol{y}_j)\,\mathrm{d}\,\Gamma_j\,\mathrm{d}\,\Gamma_i$$

$$\mathbf{M}_h[i,j] = \int_{\Gamma_i} \psi_j\,\mathrm{d}\,\Gamma_i \qquad \mathbf{q}_h[i] = \frac{\partial u}{\partial n}(\boldsymbol{x}_i)$$

where ψ are the piecewise constant basis functions For each element $u_h = \sum \psi_i u_i$ and $\mathbf{u}_h$ is given by

$$\mathbf{u}_h[i] = \begin{cases} 0 & \text{for } \Gamma_i \in \Gamma_0 \\ \tilde{u} & \text{for } \Gamma_i \in \Gamma_f \end{cases} \tag{E.9}$$

with $\tilde{u} = 1$ for problem 6.31 and $\tilde{u} = \frac{\partial u}{\partial \boldsymbol{n}}(\boldsymbol{x}_i) - Q$ for problem 6.35

Bibliography

[1] *BSEN 1078 Helmets for pedal cyclists and for users of skateboards and roller skates.* British Standards Institution, London, 1997. 115

[2] H.M. Adelman and R.T. Haftka. Sensitivity analysis of discrete structural systems. *AIAA Journal*, 24(5):823–832, 1986. 47

[3] G. Allaire, E. Bonnetier, G. Francfort, and F. Jouve. Shape optimization by the homogenization method. *Numerische Mathematik*, 76(1):27–68, 1997. 116

[4] G. Allaire, F. Jouve, and A. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194(1):363–393, 2004. 102

[5] K. Bandara, F. Cirak, G. Of, O. Steinbach, and J. Zapletal. Boundary element based multiresolution shape optimisation in electrostatics. *To be submitted*, 2013. 129, 130

[6] K. Bandara, T. Rüberg, and F. Cirak. Shape optimisation with multiresolution surfaces and immersed finite elements. *To be submitted*, 2013. 102

[7] P.K. Banerjee. *Introduction to shape optimization: theory, approximation, and computation.* McGraw-Hill, 1994. 153

[8] B. Barthelemy and R.T. Haftka. Accuracy analysis of the semi-analytical method for shape sensitivity calculation. *Mechanics of Structures and Machines*, 18(3):407–432, 1990. 75

[9] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010. 2, 17, 18

[10] M. P. Bendsøe. Optimal shape design as a material distribution problem. *Structural Optimization*, 1(4):193–202, 1989. 116

[11] M.P. Bendsøe and N. Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197–224, 1988. 116

[12] M.P. Bendsøe and O Sigmund. *Topology optimization: theory, methods, and applications*, volume 2nd Edition. Springer Verlag, 2003. 44, 118

[13] H. Biermann, D. Kristjansson, and D. Zorin. Approximate Boolean operations on free-form solids. In *SIGGRAPH 2001 Conference Proceedings*, pages 185–194. ACM Press, 2001. 51

[14] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *SIGGRAPH 2000 Conference Proceedings*, pages 113–120, 2000. 24, 73, 131

[15] K. Bletzinger, M. Firl, J. Linhard, and R. Wüchner. Optimal shapes of mechanically motivated surfaces. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):324–333, 2010. 44

[16] K.U. Bletzinger and E. Ramm. Form finding of shells by structural optimization. *Engineering with Computers*, 9(1):27–35, 1993. 5, 80, 82

[17] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1(1):1–60, 1984. 12, 17

[18] V. Braibant and C. Fleury. Shape optimal design using B-splines. *Computer Methods in Applied Mechanics and Engineering*, 44(3):247–267, 1984. 2, 44, 132

[19] A. Buffa, D. Cho, and G. Sangalli. Linear independence of the T-spline blending functions associated with some particular T-meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1437–1445, 2010. 2

[20] T.J. Cashman, N.A. Dodgson, and M. Sabin. A symmetric, non-uniform, refine and smooth subdivision algorithm for general degree B-splines. *Computer Aided Geometric Design*, 26(1):94–104, 2009. 138

[21] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput. Aided Design*, 10(6):350–355, 1978. 23

[22] J. Céa. Conception optimale ou identification de formes: calcul rapide de la dérivée directionnelle de la fonction coût. *Modélisation Mathématique et Analyse Numérique*, 20(3):371–402, 1986. 102

[23] J. Céa, S. Garreau, P. Guillaume, and M. Masmoudi. The shape and topological optimizations connection. *Computer Methods in Applied Mechanics and Engineering*, 188(4):713–726, 2000. 119

[24] E. Cervera and J. Trevelyan. Evolutionary structural optimisation based on boundary representation of NURBS. Part I: 2D algorithms. *Computers & Structures*, 83(23-24):1902–1916, 2005. 45, 108, 118

[25] G. Chaikin. An algorithm for high-speed curve generation. *Comp. Graphics and Image Processing*, 3:346–349, 1974. 18

[26] A.V. Cherkaev, Y. Grabovsky, A.B. Movchan, and S.K. Serkov. The cavity of the optimal shape under the shear stresses. *International Journal of Solids and Structures*, 35(33):4391–4410, 1998. 109

[27] K.K. Choi and S.L. Twu. On equivalence of continuum and discrete methods of shape sensitivity analysis. *AIAA Journal*, 27(10):1428–1424, 1988. 47

[28] P.G. Ciarlet. An introduction to differential geometry with applications to elasticity. *Journal of Elasticity*, 78(1):1–215, 2005. 68

[29] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998. 28

[30] F. Cirak. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *Computer-Aided Design*, 34(2):137–148, 2002. 5, 68

[31] F. Cirak and Q. Long. Subdivision shells with exact boundary control and non-manifold geometry. *International Journal for Numerical Methods in Engineering*, 88(9):897–923, 2011. 74, 146

[32] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, pages 813–833, 2001. 68

[33] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000. 2, 11, 68, 71, 73

[34] A. Cohen, I. Daubechies, and J.C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45(5):485–560, 1992. 51, 53

[35] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, 1980. 17

[36] F. Daoud, M. Firl, and K. Bletzinger. Filter techniques in shape optimization with CAD-free parametrization. In *6th World Congresses of Structural and Multidisciplinary Optimization*, 2005. 44

[37] C. De Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972. 16

[38] C. de Boor. The quasi-interpolant as a tool in elementary polynomial spline theory. *Approximation Theory*, pages 269–276, 1973. 54

[39] C. de Boor. Quasiinterpolants and approximation power of multivariate splines. In *Computation of Curves and Surfaces*. Eds. Kluwer Academic Publishers, 1990. 54

[40] P. de Casteljau. Outillages méthodes calcul. Technical report, A. Citroën, Paris, 1959. 16

[41] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *SIGGRAPH 1998 Conference Proceedings*, pages 85–94. ACM Press, 1998. 5, 21

[42] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978. 23

[43] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 1995 Conference Proceedings*, volume 6, pages 173–182. ACM Press, 1995. 27, 138

[44] K. Eppler and H. Harbrecht. Efficient treatment of stationary free boundary problems. *Applied Numerical Mathematics*, 56(10):1326–1339, 2006. 128

[45] K. Eppler and H. Harbrecht. Tracking Neumann data for stationary free boundary problems. *SIAM Journal on Control and Optimization*, 48(5):2901–2916, 2009. 129

[46] H.A. Eschenauer, V.V. Kobelev, and A. Schumacher. Bubble method for topology and shape optimization of structures. *Structural and Multidisciplinary Optimization*, 8(1):42–51, 1994. 11, 118, 122

[47] H.A. Eschenauer and N. Olhoff. Topology optimization of continuum structures: A review. *Applied Mechanics Reviews*, 54(4):331, 2001. 116, 118

[48] G.E. Farin. *Curves and surfaces for computer aided geometric design: A practical guide*. Academic Press (Boston), 1988. 17

[49] A. Finkelstein and D.H. Salesin. Multiresolution curves. In *SIGGRAPH 1994 Conference Proceedings*, pages 261–268. ACM, 1994. 48

[50] M. Flucher and M. Rumpf. Bernoulli's free-boundary problem, qualitative theory and numerical approximation. *Journal fur die reine und angewandte Mathematik*, pages 165–204, 1997. 128

[51] D.R. Forsey and R.H. Bartels. Hierarchical B-spline refinement. In *SIGGRAPH 1988 Conference Proceedings*, volume 22, pages 205–212. ACM, 1988. 40, 53

[52] M.J. Garcia and C.A. Gonzalez. Shape optimisation of continuum structures via evolution strategies and fixed grid finite element analysis. *Structural and Multidisciplinary Optimization*, 26(1-2):92–98, 2004. 104

[53] M.J. Garcia and Steven G.P. Fixed grid finite element analysis in structural design and optimisation. In *Second ISSMO/AIAA Internet Conference on Approximations and Fast Reanalysis in Engineering Optimizations*. ACM, 2000. 104

[54] M. Garland. *Quadric-based polygonal surface simplification*. PhD thesis, Georgia Institute of Technology, 1999. 32, 142

[55] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 1997 Conference Proceedings*, pages 209–216, 1997. 29, 30, 31

[56] M. Garland and Y. Zhou. Quadric-based simplification in any dimension. *ACM Transactions on Graphics (TOG)*, 24(2):209–239, 2005. 31

[57] S. Garreau, P. Guillaume, and M. Masmoudi. The topological asymptotic for PDE systems: the elasticity case. *SIAM Journal on Control and Optimization*, 39(6):1756–1778, 2001. 118, 120, 150

[58] S.J. Gortler and M.F. Cohen. Hierarchical and variational geometric modeling with wavelets. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 35–ff. ACM, 1995. 49, 53, 54

[59] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *SIGGRAPH 1999 Conference Proceedings*, pages 325–334, 1999. 33, 37, 38, 62, 64

[60] S. Ha, K.K. Choi, and S. Cho. Numerical method for shape optimization using t-spline based isogeometric method. *Structural and Multidisciplinary Optimization*, 42(3):417–428, 2010. 5

[61] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910. 48

[62] R.T. Haftka and H.M. Adelman. Recent developments in structural sensitivity analysis. *Structural and Multidisciplinary Optimization*, 151:137–151, 1989. 47

[63] R.T. Haftka and R.V. Garandhi. Structural shape optimization–A survey. *Computer Methods in Applied Mechanics and Engineering*, 57(1):91–106, 1986. 4, 44, 132

[64] R.T. Haftka and Z Gürdal. *Elements of Structural Optimization*, volume 11 of *Solid Mechanics and its Applications*. Kluwer Academic Publishers, 1992. 76

[65] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *of the 20th annual conference on*, pages 35–44, 1993. 51

[66] J. Haslinger and R. A. E. Mäkinen. *Introduction to shape optimization: theory, approximation, and computation*, volume 7. SIAM, 2003. 44, 128

[67] P.S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, Carnegie Mellon University, 1997. 8, 28

[68] C.M. Hoffmann. *Geometric and solid modeling: an introduction*. Morgan Kaufmann Publishers Inc., 1989. 100

[69] H. Hoppe. Progressive meshes. In *SIGGRAPH 1996 Conference Proceedings*, pages 99–108. ACM, 1996. 8, 28, 33

[70] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH 1994 Conference Proceedings*, pages 295–302. ACM Press, 1994. 51, 55

[71] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH 1993 Conference Proceedings*, pages 19–26. ACM, 1993. 55

[72] G.C. Hsiao and W.L. Wendland. *Boundary integral equations*. Springer, 2008. 153

[73] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, 2005. 1, 2, 68

[74] M.H. Imam. Three-dimensional shape optimization. *International Journal for Numerical Methods in Engineering*, 18(5):661–673, 1982. 44

[75] K. Ito, K. Kunisch, and G.H. Peichl. Variational approach to shape derivatives for a class of Bernoulli problems. *Journal of Mathematical Analysis and Applications*, 314(1):126–149, 2006. 128

[76] S.G. Johnson. The nlopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt. 57, 78, 130

[77] J. Kiendl, Y. Bazilevs, M.C. Hsu, R. Wüchner, and K.U. Bletzinger. The bending strip method for isogeometric analysis of Kirchhoff–Love shell structures comprised of multiple patches. *Computer Methods in Applied Mechanics and Engineering*, 199(37):2403–2416, 2010. 68

[78] J. Kiendl, K.U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3902–3914, 2009. 68

[79] J.M. Kiendl. *Isogeometric analysis and shape optimal design of shell structures*. Shaker-Verlag, 2011. 45

[80] Y.Y. Kim and G.H. Yoon. Multi-resolution multi-scale topology optimization – a new paradigm. *International Journal of Solids and Structures*, 37, 2000. 45

[81] L. Kobbelt, S. Campagna, and H. Seidel. A general framework for mesh decimation. In *Graphics Interface*, pages 43–50, 1998. 31

[82] L. Kobbelt, S. Campagna, J. Vorsatz, and H.P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH 1998 Conference Proceedings*, pages 105–114, New York, New York, USA, 1998. ACM Press. 33, 35, 37, 40, 60

[83] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH 1996 Conference Proceedings*, pages 313–324. ACM, 1996. 27

[84] E.S. Kristensen and N.F. Madson. On the optimum shape of fillets in plates subjected to multiple in-plane loading cases. *International Journal for Numerical Methods in Engineering*, 10:1007–1019, 1976. 107

[85] S. Lanquetin and M. Neveu. Reverse Catmull-Clark subdivision. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision(WSCG)*, pages 319–326, 2006. 51

[86] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *SIGGRAPH 2000 Conference Proceedings*, pages 85–94. ACM Press, 2000. 27

[87] A.W.F. Lee, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 1998 Conference Proceedings*, pages 95–104, 1998. 27, 138

[88] N. Litke, A. Levin, and P. Schröder. Fitting subdivision surfaces. *Proceedings Visualization, VIS '01.*, pages 319–568, 2001. 51, 55

[89] E.H. Lockwood. *A book of curves.* Cambridge University Press, 1971. 78

[90] Q. Long. *Subdivision finite elements for geometrically complex thin and thick shells.* PhD thesis, University of Cambridge, 2009. 74

[91] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987. 23

[92] M. Lounsbery, T.D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics (TOG)*, 16(1):34–73, 1997. 48

[93] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III (2002)*, pages 113–134, 2002. 130

[94] N.J. Mills and A. Gilchrist. Finite-element analysis of bicycle helmet oblique impacts. *International Journal of Impact Engineering*, 35(9):1087–1101, 2008. 115

[95] B. Mohammadi, O. Pironneau, B. Mohammadi, and O. Pironneau. *Applied shape optimization for fluids*, volume 28. Oxford University Press, Oxford, 2001. 44

[96] T.H. Nguyen, G.H. Paulino, J. Song, and C.H. Le. A computational paradigm for multiresolution topology optimization (MTOP). *Structural and Multidisciplinary Optimization*, 41(4):525–539, 2009. 45

[97] K. Noboru, K.Y. Chung, T. Toshikazu, and J.E. Taylor. Adaptive finite element methods for shape optimization of linearly elastic structures. *Computer Methods in Applied Mechanics and Engineering*, 57(1):67–89, 1986. 108

[98] J. Norato, R. Haber, D. Tortorelli, and M.P. Bendsøe. A geometry projection method for shape optimization. *International Journal for Numerical Methods in Engineering*, 60(14):2289–2312, 2004. 108

[99] A.A. Novotny, R.A. Feijóo, E. Taroco, and C. Padra. Topological sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering*, 192(7-8):803–829, 2003. 120, 149, 150

[100] A.A. Novotny, R.A. Feijóo, E. Taroco, and C. Padra. Topological-shape sensitivity method: theory and applications. *Optimization*, pages 469–478, 2006. 120

[101] A.A Novotny, R.A. Feijóo, E. Taroco, and C. Padra. Topological sensitivity analysis for three-dimensional linear elasticity problem. *Computer Methods in Applied Mechanics and Engineering*, 196(41-44):4354–4364, 2007. 120, 150

[102] G. Of, O. Steinbach, and W.L. Wendland. The fast multipole method for the symmetric boundary integral formulation. *IMA J. Numer. Anal.*, 26(2):272–296, 2006. 11, 153

[103] N. Olhoff and J. Rasmussen. Study of inaccuracy in semi-analytical sensitivity analysis – a model problem. *Structural and Multidisciplinary Optimization*, 3(4):203–213, 1991. 75

[104] L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. *Computer-Aided Design*, 19(9):485–498, 1987. 15

[105] L. Piegl and W. Tiller. *The NURBS Book*. Monographs in Visual Communication. Springer-Verlag:New York, 2nd ed. edition, 1997. 13

[106] T.A. Poulsen. Topology optimization in wavelet space. *International Journal for Numerical Methods in Engineering*, 53(3):567–582, 2002. 45

[107] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*. Springer, 2002. 13

[108] X. Qian. Topology optimization in B-spline space. *Computer Methods in Applied Mechanics and Engineering*, 265(0), 2013. 45

[109] T. Rüberg and F. Cirak. An immersed finite element method with integral equation correction. *International Journal for Numerical Methods in Engineering*, 86(1):93–114, 2011. 104, 152

[110] T. Rüberg and F. Cirak. Subdivision-stabilised immersed B-spline finite elements for moving boundary flows. *Computer Methods in Applied Mechanics and Engineering*, 209:266–283, 2011. 11, 104, 152

[111] M. Sabin. Recent progress in subdivision: a survey. *Advances in Multiresolution for Geometric Modelling*, 2005. 23

[112] P. Schröder. Wavelets in computer graphics. *Proceedings of the IEEE*, 84(4):615–625, 1996. 48

[113] P. Schröder. Subdivision, multiresolution and the construction of scalable algorithms in computer graphics. *Multivariate Approximation and Applications*, pages 213–251, 2001. 50, 54

[114] A. Schumacher. Parameter-based topology optimization for crashworthiness structures. *Structural and Multidisciplinary Optimization*, 2005. 118

[115] T. W. Sederberg, G. T. Finnigan, X. Li, H Lin, and H. Ipson. Watertight trimmed NURBS. *ACM Transactions on Graphics*, 27(3):1, 2008. 17

[116] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *ACM Transactions on Graphics*, volume 22, pages 477–484, 2003. 2

[117] Y. Seo, H. Kim, and S. Youn. Shape optimization and its extension to topological design based on isogeometric analysis. *International Journal of Solids and Structures*, 47(11-12):1618–1640, 2010. 5, 118

[118] Y.D. Seo, H.J. Kim, and S.K. Youn. Isogeometric topology optimization using trimmed spline surfaces. *Computer Methods in Applied Mechanics and Engineering*, 199(49-52):3270–3296, 2010. 118

[119] O. Sigmund and J. Petersson. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural Optimization*, 16(1):68–75, 1998. 44

[120] J. Sokolowski and A. Zochowski. On topological derivative in shape optimization. Technical report, INRIA, 1997. 119

[121] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH 1998 Conference Proceedings*, pages 395–404, 1998. 144, 146

[122] E.J. Stollnitz, A.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: a primer. 1. *Computer Graphics and Applications, IEEE*, 15(3):76–84, 1995. 48

[123] K. Svanberg. The method of moving asymptotes — a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987. 57

[124] K. Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, 12:555–573, 2002. 57, 78, 130

[125] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 1995 Conference Proceedings*, pages 351–358. ACM Press, 1995. 35, 61

[126] W. Tiller. Rational B-splines for curve and surface representation. *Computer Graphics and Applications, IEEE*, 3(6):61–69, 1983. 15

[127] K.J. Versprille. *Computer-aided design applications of the rational B-spline approximation form.* PhD thesis, Syracuse University, 1975. 15

[128] W. Wall, M. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2976–2988, 2008. 5

[129] R.J. Yang and M.E. Botkin. Comparison between the variational and implicit differentiation approaches to shape design sensitivities. *AIAA Journal*, 24(6):1027–1032, 1986. 47

[130] J.P. Zolesio and M.C. Delfour. Shapes and geometries. *Analysis, differential calculus and optimization, SIAM, Philadelphia*, 2001. 129

[131] D. Zorin and P. Schröder. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*, 2000. 18, 23, 141

[132] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH 1997 Conference Proceedings*, pages 259–268. ACM Press/Addison-Wesley Publishing Co., 1997. 37, 138