Isogeometric Design, Analysis and Optimisation of Lattice-Skin Structures

Xiao Xiao

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Jesus College   October 2018

Declaration

Xiao Xiao
October 2018

Acknowledgements

I would like first to express my special thanks to my PhD supervisor, Dr. Fehmi Cirak, for his full support and guidance during my doctoral research. All the suggestions that he provided during my PhD life are very much appreciated. More importantly, his dedication to research has inspired me to learn from him how to be a professional researcher.

I would also like to thank Dr Malcolm Sabin for his kind and patient guidance to help me enter the research field of the CAD. Without his help I wouldn't have gained the knowledge in the CAD so quickly.

My advisor, Prof. Allan McRobie, provided valuable feedback when I started my doctoral reseach which is also greatly appreciated.

I would like to thank all the group members whom I have worked with, Dr Matija Kecman, Dr Musabbir Majeed, Dr Hong Xiao, Qiaoling Zhang, Ge Yin, Eky Valentian Febrianto, Arion Pons and Sumudu Herath, for giving me a warm and happy lab life in the past four years.

The China Scholarship Council (CSC) studentship is appreciated to enable me to conduct doctoral research work in Cambridge.

I would like to give my final gratitude to my parents who have been always there to support me. Many thanks for their endless encouragement and understanding to help me focus on and finally finish the doctoral research.

Abstract

The advancements in additive manufacturing techniques enable novel designs using lattice structures in mechanical parts, lightweight materials, biomaterials and so forth. Lattice-skin structures are a class of structures that couple thin-shells with lattices, which potentially combine the advantages of the thin-shell and the lattice structure. A new and systematic isogeometric analysis approach that integrates the geometric design, structural analysis and optimisation of lattice-skin structures is proposed in the dissertation.

In the geometric design of lattice-skin structures, a novel shape interrogation scheme for splines, specifically subdivision surfaces, is proposed, which is able to compute the line/surface intersection efficiently and robustly without resorting to successive refinements or iterations as in Newton-Raphson method. The line/surface intersection algorithm involves two steps: intersection detection and intersection computation. In the intersection detection process, a bounding volume tree of k-dops (discrete oriented polytopes) for the subdivision surface is first created in order to accelerate the intersection detection between the line and the surface. The spline patches which are detected to be possibly intersected by the line are converted to Bézier representations. For the intersection computation, a matrix-based algorithm is applied, which converts the nonlinear intersection computation into solving a sequence of linear algebra problems using the singular value decomposition (SVD). Finally, the lattice-skin geometry is generated by projecting selected lattice nodes to the nearest intersection points intersected by the lattice edges. The Stanford bunny example demonstrates the efficiency and accuracy of the developed algorithm.

The structural analysis of lattice-skin structures follows the isogeometric approach, in which the thin-shell is discretised with spline basis functions and the lattice structure is modelled with pin-jointed truss elements. In order to consider the lattice-skin coupling, a Lagrange multiplier approach is implemented to enforce the displacement compatibility between the coupled lattice nodes and the thin-shell. More importantly, the parametric coordinates of the coupled lattice nodes on the thin-shell surface are obtained directly from the lattice-skin geometry generation, which integrates the design and analysis process of lattice-skin structures. A sandwich plate example is analysed to verify the implementation and the accuracy of the lattice-skin coupling computation.

In addition, a SIMP-like lattice topology optimisation method is proposed. The topology optimisation results of lattice structures are analysed and compared with several examples adapted from the benchmark examples commonly used in continuum topology optimisation. The SIMP-like lattice topology optimisation proposed is further applied to optimise the lattice in lattice-skin structures. The lattice-skin topology optimisation is fully integrated with the lattice-skin geometry design since the sensitivity analysis in the proposed method is based on lattice unit cells which are inherited from the geometry design stage.

Finally, shape optimisation of lattice-skin structures using the free-form deformation (FFD) technique is studied. The corresponding shape sensitivity of lattice-skin structures is derived. The geometry update of the lattice-skin structure is determined by the deformation of the FFD control volume, and in this process the coupling between lattice nodes and the thin-shell is guaranteed by keeping the parametric coordinates of coupled lattice nodes which are obtained in the lattice-skin geometry design stage. A pentagon roof example is used to explore the combination of lattice topology optimisation and shape optimisation of lattice-skin structures.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation and objectives

Lattice structures and additive manufacturing Lattice structures are widely used in various engineering fields, from common structural forms in architecture to microstructured lightweight materials. An advantage of lattice structures is that they can be assembled to create a structural hierarchy. For example, the Eiffel Tower is a pioneer project using a hierarchical lattice structure of three orders as shown in Figure 1.1, making its relative density (ratio of mass density to material density) only 1.2×10^{-3} times that of cast iron [1]. Also, it is flexible to adapt the configuration and pattern of lattice structures in order to conform to free-form surfaces, since surfaces can be approximated with triangles, quadrilaterals and other polygonal shapes. For these reasons lattice structures are commonly used in architecture with free-form surfaces.

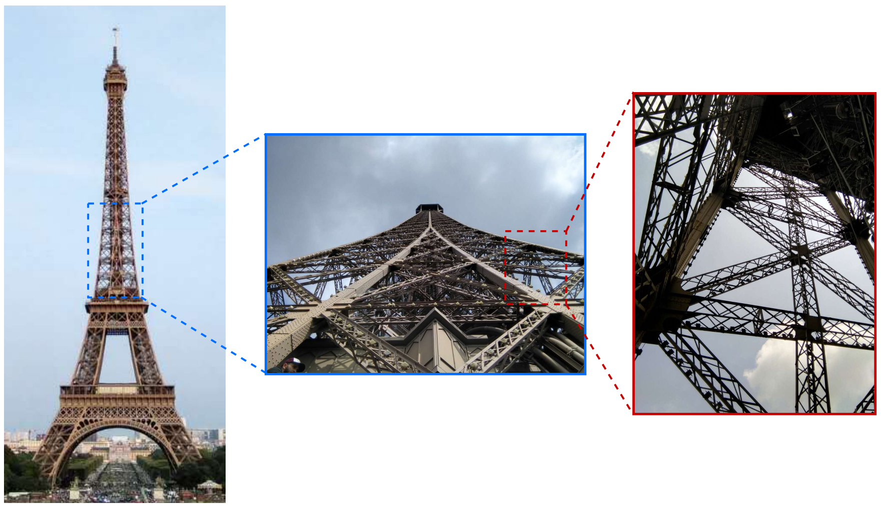
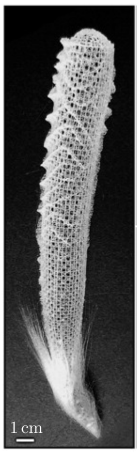
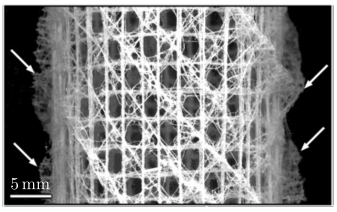


Fig. 1.1 Hierarchical lattice structure of the Eiffel Tower.

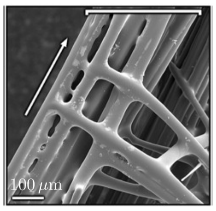
Figure 1.2 shows the skeleton of the Euplectella sponge comprised of a hierarchical lattice structure, which is one of the many examples of lattice structures in nature. The surface of the sponge is a square-grid lattice comprised of vertical and horizontal struts reinforced with diagonal elements, and each strut consists of a bundle of sponge spicules [2]. Inspired by nature, lightweight cellular materials with lattice structures are designed for various purposes, for example, thermal insulation, acoustic absorption and energy damping etc. [3]. More applications of cellular materials can be found in, for example, biomaterials and tissue engineering [4].



(a) Sponge



(b) Square-grid lattice comprised of vertical and horizontal struts with diagonal elements

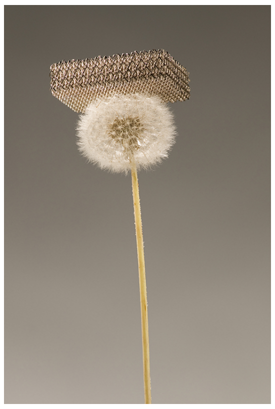


(c) Close-up view of a strut comprised of bundles of sponge spicules

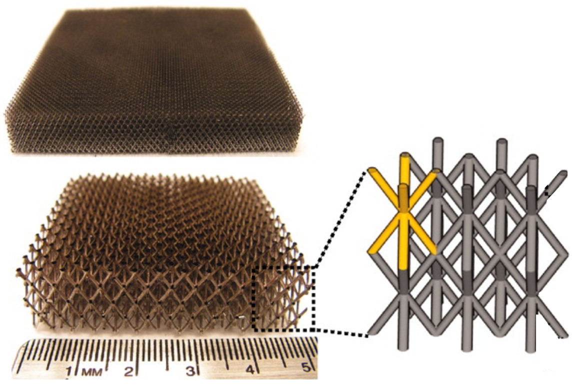
Fig. 1.2 The hierarchical lattice skeleton of the Euplectella sponge. [1]

The development of additive manufacturing techniques makes it possible to fabricate lattice structures with element sizes varying from metres to nanometres, which expands the application of lattice structures extensively. Several different additive manufacturing techniques have been developed for fabricating lattice structures in micro and nano scales, for example inkjet printing, fused deposition modelling, selective laser sintering, stereolithography, etc. [5, 6]. At present materials like polymers, metals and ceramics [7–9] have been successfully applied in the additive manufacturing. Figure 1.3 shows an example of a lightweight microlattice structure fabricated with polymer using stereolithography. The microlattice is assembled with periodic lattice unit cells, and the smallest resolution of the lattice materials can be 100 nm. It was reported by Schaedler *et al.* [7] that the volumetric mass density of this microlattice material can be $0.9\,\mathrm{mg/cm^3}$ (not including the air weight in the pores), which is even smaller than the density of air which is approximately $1.225\,\mathrm{mg/cm^3}$.

[1]Figure 1.2 is adapted from Aizenberg *et al.* [2].



(a) Lightweight microlattice



(b) Microlattice structure

Fig. 1.3 A lightweight microlattice material comprised of lattice unit cells. [2]

In contrast to conventional subtractive manufacturing, additive manufacturing can handle complex geometries without significant extra effort. Several additive manufacturing techniques can deal with metals, for example, techniques in categories of powder bed fusion, binder jetting and direct energy deposition [10]. New creative structural forms in mechanical engineering can then be designed by combining with lattice structures. A benefit of adding lattice structures to replace a solid region in a mechanical part is that the reduced weight of the mechanical part can potentially improve its other capabilities and can significantly reduce the time needed in the additive manufacturing. Figure 1.4 shows two examples of mechanical parts with lattice structures manufactured by additive manufacturing. The bracket shown in Figure 1.4a is fabricated with titanium, and it combines lattice structures with topology optimisation of the solid aiming to minimise the component weight. The connecting part shown in Figure 1.4b replaces its solid part with lattice structures such that the weight of the component is reduced while still satisfying the mechanical requirement.

It can be concluded that the lattice structure is a promising option in the engineering design with the development of additive manufacturing. Using lattice structures can achieve equivalent or comparable mechanical properties with much less material, which is quite appealing in industrial design, especially when the weight of structures has a significant influence on the overall performance of the design, for example, in automobile and aerospace industries. In addition, lattice structures are flexible to design and have advantages in conforming to free-form surfaces. If the lattice structure contains unit cells, it can be periodically assembled, and this is often used in material fabrication.

[2]Figure 1.3 is adapted from HRL Laboratories and Schaedler *et al.* [7].

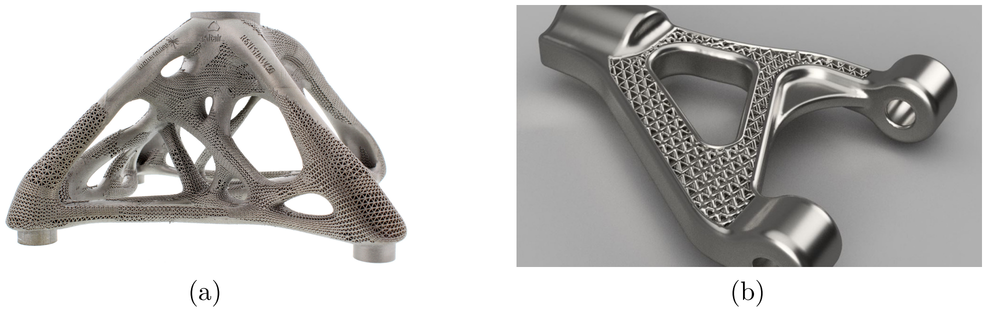


Fig. 1.4 Infilled lattice structures in mechanical parts. [3]

When lattice structures are used as lattice materials, different configurations of the lattice structure can result in significantly different material properties, for example the stretching-dominated lattice structure in lightweight stiff materials and the bending-dominated lattice structure in foam materials. This feature provides great flexibility to fabricate lattice materials with unique features by manipulating the configuration of lattice structures.

Integrated isogeometric design, analysis and optimisation The term isogeometric analysis was coined by Hughes et al. [11], which is a computational method that combines the finite element method with geometric representations used in the computer-aided design. There are a number of earlier works on using unified representations for geometries and finite element analysis, for example, uniform B-spline representations in [12] and subdivision surfaces in [13, 14]. Much research work on isogeometric analysis has been reported since then, including its applications in structural analysis [11, 15–17], shape optimisation [18–20], fluid-structure interaction [21–23], and so forth.

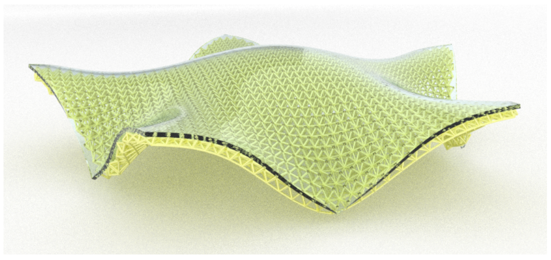
In contrast to the conventional finite element analysis which treats the CAD geometry and the finite element model in separate modules, the isogeometric analysis bridges the gap between the CAD geometry model and the finite element model. This property is highly favourable as the mesh generation would account for a large portion of the total computational time in conventional finite analysis. It was reported that the ratio of time needed by modelling and analysis is 80/20 in general [24], and it becomes worse if the mesh generated cannot be guaranteed to be analysis-suitable for models with complex geometries.

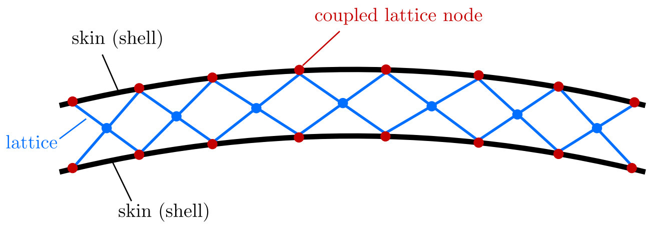
[3]Figure 1.4 is downloaded from http://resources.renishaw.com and https://www.autodesk.com

Several geometric representations that are commonly used in isogeometric analysis include the non-uniform rational B-splines (NURBS) [11], T-splines [25–27] and subdivision surfaces [13, 15, 28]. NURBS has gained great attention in the isogeometric analysis since it is the predominant geometry representation in most commercial CAD packages. However, the problem with NURBS is that gaps can exist between NURBS patches especially when trimmed NURBS patches are presented in the geometry model, which is unadvoidable in most engineering designs as multiple NURBS surfaces are usually needed to model a freeform shape. T-splines and subdivision surfaces are widely used to generate watertight smooth surfaces. T-splines can be seen as a generalisation of NURBS but having local refinement property. The control polygon of a surface can be locally refined with T-splines by adding T-junctions [29]. Compared with a NURBS model, a T-spline model of the same geometry can have much fewer control points since the T-spline does not have the strict topological constraint as the NURBS, which is another advantage for the geometric design. Analysis-suitable T-splines was proposed in [30, 31] in order to deal with the issue of T-splines that the basis functions cannot be guaranteed to be linearly independent [32]. Subdivision surfaces can be seen as a generalisation of spline surfaces [33] and it has been widely adopted in the animation industry. The underlying basis functions of subdivision surfaces have the mathematical properties of partition unity and linear independence, which ensures that they can be used directly for the finite element analysis. In fact, the NURBS-compatible subdivision surfaces have been studied [34, 35], and subdivision surfaces are becoming a promising option in some CAD packages.

Using isogeometric analysis has a natural advantage in structural optimisation because of its integration of geometry and analysis models. In structural optimisation problems design variables are updated after each optimisation iteration. The design variables can be sizes of structural members, nodal coordinates and connectivity in the structure. All these changes in geometry require a mesh update for the conventional finite element analysis, which can be time-consuming and error-prone. Since the geometry model is integrated with the finite element model in the isogeometric analysis, the updated geometry model after each optimisation process can be used directly as the finite element model without remeshing.

Objectives The aim of the research is to explore a new and systematic approach to integrate the geometric design, structural analysis and optimisation of lattice-skin structures in which the lattice structure and the thin-shell are coupled. Figure 1.5 is an example of the lattice-skin structure. As described in preceding sections, lattice structures are ideal for multi-functional design, for example, for lightweight, stiffness, heat transfer and so forth. With the advances of the additive manufacturing, it is becoming increasingly possible to design with lattice structures and create novel and functional products. In addition, thin-shell structures exhibit low weight and superior load carrying capacity and they are widely used in large-span designs. It is promising that lattice-skin structures can combine the advantages of lattice structures and thin-shells.


(a) A lattice-skin structure



(b) Sketch of the lattice-skin structure

Fig. 1.5 Illustration of the lattice-skin structure.

However, it is in general not feasible to model the entire lattice-skin structure in a CAD (Computer-Aided Design) system due to the large number of geometry entities required and the poor scalability of CAD systems [36, 37], although the size of lattices that can be represented in the CAD system is increasing continuously. If the lattice is modelled as a solid in the CAD system, the Boolean operations required for the solid modelling of a large number of lattice struts are prohibitively expensive to compute [38, 39]. In practice, this limitation is overcome by designing only the skin in the CAD system, and the skin surface is triangulated before adding the lattice outside the CAD system. Nonetheless, this approach leads to accuracy issues, especially when the lattice-skin structure generated is analysed with the finite element method.

In order to cope with these issues in the design and analysis of lattice-skin structures, the objectives of the presented research on lattice-skin structures are as follows.

- First of all, a robust and efficient approach is needed for the shape interrogation of spline surfaces in order to combine the lattice with the spline surface used in CAD systems directly to generate lattice-skin structures. This process involves computing the intersection of the lattice and the spline surface, and algebraic geometry techniques are developed to compute the intersection points.

- Secondly, the coupling between the lattice structure and the thin-shell needs to be considered in the structural analysis of the lattice-skin structure. A numerical computation scheme is implemented to consider the lattice-skin coupling.

- In addition, lattice topology optimisation is applied to the lattice-skin structure. Though lattice structures are usually used for the lightweight purpose, the optimisation of lattice structures is still needed to make use of the material more efficiently. To this end, a lattice topology optimisation method is developed specifically for lattice structures comprised of unit lattice cells and then applied to lattice-skin structures.

- Finally, shape optimisation of the lattice-skin structure is studied. Although the shape optimisation methods of lattice structures and thin-shells have been studied extensively, they cannot be applied directly to the lattice-skin structure, as the lattice and the thin-shell skin have different geometry parameterisations and the lattice-skin coupling should also be considered in shape optimisation.

1.2 Methodology

Shape interrogation of spline surfaces The process of extracting information from geometries is commonly referred to as *shape interrogation* in the computer-aided design literature. In order to generate and analyse lattice-skin structures, the intersection between lattice struts and the thin-shell needs to be computed. The physical coordinates of intersection points are needed for the geometry generation, and the corresponding parametric coordinates on the thin-shell surface are required for the lattice-skin coupling analysis. The line/surface intersection computation involves the shape interrogation of spline surfaces. Some approaches commonly used for the shape interrogation of spline surfaces, including the triangulation, the marching approach, the subdivision approach and the algebraic approach, are summarised as follows.

The most straightforward approach is the triangulation of spline surfaces [40, 41], that is, the spline surface is approximated by piecewise linear triangles. The triangulation is a standard geometric representation in additive manufacturing which in general accepts STL geometry models for generating printable files. After the spline surface is triangulated, the line/surface intersection is reduced to find the intersection between a line and a collection of triangles which becomes straightforward to solve. Though this approach is easy to implement and the robustness can be guaranteed, the accuracy of intersection computation depends on how densely the surface is triangulated. It is probable to have a huge number of triangles in the geometry model in order to obtain the intersection points within a small tolerance. Furthermore, the parametric coordinates of intersection points on the surface are not straightforward to be inferred from the intersection computation.

The marching approach takes advantage of the geometric information of the spline surface [42, 43], for example, the surface derivatives evaluated at points on the spline surface. Starting points on the line and the spline surface are required to initialise the marching process. The marching process involves a major step and several minor steps. At each step, the difference between the point on the line and the point on the surface is approximated linearly which involves the first derivatives evaluated at those points. The increments of parameters are obtained by solving a linear approximation equation system. The iteration process terminates when the difference between the point on the line and the point on the surface is within a tolerance. This process is essentially the Newton-Raphson method and has a quadratic convergence, though the robustness of finding a solution cannot be guaranteed which depends on the selection of starting points.

The divide-and-conquer approach, also known as the subdivision approach, involves two steps in each iteration, intersection detection and surface refinement. In general, the spline surface has the convex hull property since the basis functions are non-negative, that is, the surface lies entirely within the convex hull of its control points. Hence, the intersection between the line and the surface can be detected (there is a possibility that the line intersects with the spline surface) if the line intersects with the convex hull of the surface. In other words, if there is no intersection between the line and the convex hull of the surface, the line cannot intersect with the surface. However, the convex hull of the spline surface is not cheap to compute. Alternatively, the bounding box and the k-dop (discrete orientation polytope) of the spline surface can be used for the intersection detection, which are easy to compute though not as tight as the convex hull. After a possible intersection between the line and the spline surface is detected, the spline surface is refined into smaller pieces, and the intersection detection is performed again to the line and the refined surface pieces. This process continues until the pieces can be approximated by planes, and the intersection computation is reduced to the simple line/plane intersection problem.

The algebraic geometry approach is a class of methods that aim to obtain the implicitisation of the spline surface [44]. As a matter of fact, every parametric polynomial surface can be represented in implicit form. The classical algebraic geometry approach is mainly about the elimination of variables in polynomial equations that represent the spline surface [45]. The resultant of polynomials is the key tool in these elimination methods, which determines if there is any common root in two polynomials. Two polynomials have common roots if and only if the resultant is zero. The resultant forms commonly used include the Sylvester's resultant and the Bézout's resultant [44, 46]. For planar curves the elimination methods have been well understood. The moving line method proposed by Sederberg et al. [47, 48] gives a geometric interpretation of the implicitisation for planar curves. The basic idea of the moving line method is based on the fact that any conic curve can be generated by the intersection of two pencils of lines [49]. For example, a quadratic planar curve can be generated by the intersection of two linear pencils, and a cubic planar curve can be generated by the intersection of a linear pencil and a quadratic pencil. The moving line method gives a matrix which is equivalent to the Bézout matrix in a more compact way.

However, the traditional elimination methods cannot be extended straightforwardly to spline surfaces, as the surface involves more geometric information and the base points [4] are not as easy to cope with as for planar curves [46]. In addition, the matrix obtained cannot always be non-singular, which is the requirement if the determinant of the matrix is used to obtain the implicit equation of the surface. In order to avoid the requirement of the non-singular matrix, Busé et al. [50, 51] proposed the implicit matrix representations of rational surfaces, in which the surface is implicitised with matrix form instead of a polynomial equation as in traditional algebraic geometry approach. Once the implicit matrix representation of the spline surface is obtained, whether or not a point in space is on the surface can be determined by evaluating the rank of the implicit matrix after substituting the coordinates of the point into the matrix. The point is on the surface if and only if the rank of the corresponding implicit matrix drops. The line/surface intersection can then be computed by solving a generalised eigenvalue problem and a few small singular value decompositions.

Topology optimisation The topology optimisation of continuum structures has been studied extensively [52, 53] with approaches dealing with the material distribution in the design domain. The homogenisation method proposed by Bendsøe et al. [54, 55] uses intermediate material densities in the problem formulation such that it contains microscopic structures with the optimised material densities in the final result. The macroscopic material property of the microstructure is required, which can be obtained

[4]For a surface parametrised by $\boldsymbol{x} = (x^1(\boldsymbol{\theta}, x^2(\boldsymbol{\theta}), x^3(\boldsymbol{\theta}))^\mathsf{T} = (\frac{f_1(\boldsymbol{\theta})}{f_4(\boldsymbol{\theta})}, \frac{f_2(\boldsymbol{\theta})}{f_4(\boldsymbol{\theta})}, \frac{f_3(\boldsymbol{\theta})}{f_4(\boldsymbol{\theta})})^\mathsf{T}$, the base points are the points $\boldsymbol{\theta}$ where f_1, f_2, f_3 and f_4 all vanish simultaneously [46].

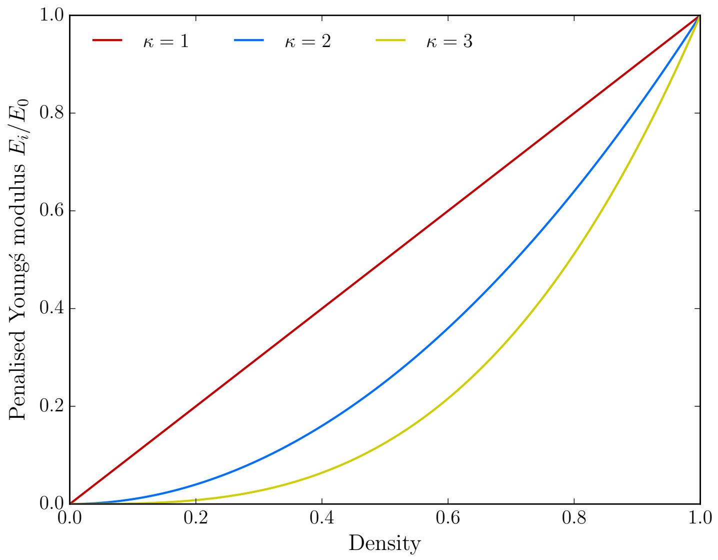


Fig. 1.6 Penalisation of Young's modulus in the SIMP method.

with the homogenisation theory [56]. The main drawback of this method is that it is not straightforward to get the optimal microstructure corresponding to an intermediate material density. This issue can be alleviated by considering only a subclass of microstructures.

The solid isotropic material with penalisation (SIMP) method considers intermediate material densities between solid and void with a penalisation [55, 57]. For the optimisation problem of structural compliance, the Young's modulus of the material can be penalised as follows,

$$E_i = \rho_i^\kappa E_0 \,, \tag{1.1}$$

where κ is the penalty parameter; ρ denotes the density of the material, which is between 0 (void) and 1 (solid); E_0 is the Young's modulus of the solid material. The lower bound of the density ρ takes a small value, e.g. 10^{-4}, in order to avoid the singularity of the stiffness matrix. The Young's modulus penalised with (1.1) instead of the original solid Young's modulus is used in the finite element analysis as well as in the sensitivity analysis for the gradient-based optimisation.

The penalisation of Young's modulus using (1.1) is illustrated in Figure 1.6. When the penalty parameter $\kappa > 1$, the Young's modulus is penalised such that it is disproportionally low between the void state ($\rho = 0$) and the solid state ($\rho = 1$). Therefore, the material density ρ will either approach to 0 or 1 in the optimal solution; in other words, intermediate density values will be avoided in the optimisation as the material is not used efficiently. In addition, as the penalty parameter becomes larger, the percentage of intermediate densities will become smaller in the optimised structure. Nonetheless, a large penalty parameter can lead to numerical instability.

A modified SIMP method is formulated as [58]

$$E_i = E_{\min} + \rho_i^{\kappa}(E_0 - E_{\min})\,, \tag{1.2}$$

where $E_{\min}$ is the Young's modulus of the void material, which is set as a small value, e.g. 10^{-4}. The advantage of using this modified formulation is that the density ρ can be zero and the minimum Young's modulus is independent of the penalty parameter.

In order to deal with numerical instabilities, for example chequerboard patterns, mesh dependencies and local minima, in topology optimisation using the SIMP method, filtering techniques have been employed which include density filtering [59, 60] and sensitivity filtering [61, 58]. In the density filtering approach, the element density is filtered with

$$\tilde{\rho}_e = \frac{\sum_{i\in\mathcal{N}_e} w(\boldsymbol{x}_i)\rho_i V_i}{\sum_{i\in\mathcal{N}_e} w(\boldsymbol{x}_i) V_i}\,, \tag{1.3}$$

where $\boldsymbol{x}_i$ denotes the position of the i-th element which is usually represented by its centroid; V_i is the volume of the i-th element; the weight $w(\boldsymbol{x}_i)$ can be determined by a linear decaying function

$$w(\boldsymbol{x}_i) = \max\left(R_f - ||\boldsymbol{x}_i - \boldsymbol{x}_e||,\, 0\right) \tag{1.4}$$

with R_f a prescribed filtering range and $\boldsymbol{x}_e$ the position of an element in the set $\mathcal{N}_e$. $\mathcal{N}_e$ denotes the neighbouring elements of the i-th element which are located within the range R_f, and $||\cdot||$ denotes the Euclidean distance between two points.

The idea of sensitivity filtering is that the filtered sensitivity instead of the real sensitivity is used in the gradient-based optimisation. This can be easily implemented with, according to [62],

$$\widetilde{\frac{\partial J}{\partial \rho_e}} = \frac{\sum_{i\in\mathcal{N}_e} w(\boldsymbol{x}_i)\rho_i \frac{\partial J}{\partial \rho_i}}{\sum_{i\in\mathcal{N}_e} w(\boldsymbol{x}_i)\rho_i} \tag{1.5}$$

with the same symbols as used in the density filtering (1.3). Recently, some attempts have been made to interpret the sensitivity filtering from the continuum mechanics perspective. The sensitivity filtering concept used for the compliance optimisation problem can be derived from the strain energy minimisation using a nonlocal elasticity formulation [63].

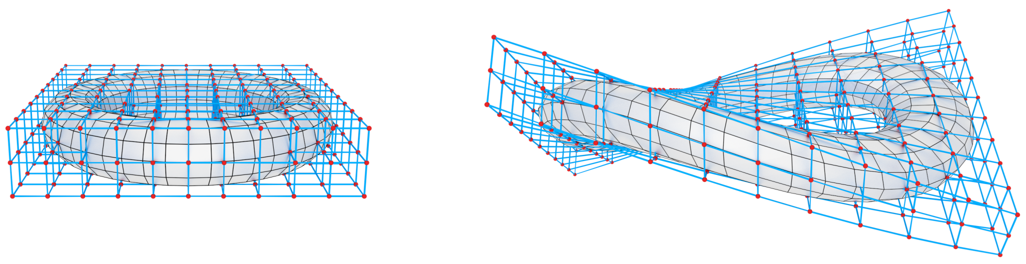
In lattice topology optimisation of lattice-skin structures, a SIMP-like method is proposed in order to obtain a topology optimised lattice structure in the lattice-skin structure. Since the lattice structure is discrete and the concept of material density in topology optimisation of the continuum structure cannot be applied directly to the lattice structure, the SIMP method used in the continuum structure is adapted as follows. The element stiffness of a lattice strut can be written in the following equivalent way,

$$\mathbf{k}_i = \frac{A_i}{A_i^{\min}} \mathbf{k}_i^{\min} = \rho_i \mathbf{k}_i^{\min} \,, \tag{1.6}$$

where A_i and $A_i^{\min}$ are the cross-sectional area and the prescribed minimum cross-sectional area of the i-th strut respectively, and $\mathbf{k}_i^{\min}$ is the element stiffness corresponding to $A_i^{\min}$. The ratio ρ_i is modified if A_i is less than $A_i^{\min}$. The detailed modification is described in Chapter 5. It becomes lattice topology optimisation when $A_i^{\min}$ takes the original cross-sectional area of the strut A_i^0.

Shape optimisation with free-form deformation An intuitive way for shape optimisation is to take nodal coordinates of the geometry as design variables. Nevertheless, the final optimal shape obtained in this way may be deformed severely so that the shape cannot be used in practice. The problem of using nodal coordinates as design variables directly was also stated in [64]. For geometries represented with subdivision surfaces, a multiresolution shape optimisation scheme was proposed by Bandara et al. [19, 20] to deal with this problem when nodal coordinates are considered as design variables, in which the structural analysis and the geometry update are performed on different subdivision levels. The structural analysis is performed on a fine level, and the gradient obtained on this fine level is projected back to the coarse level in order to update the coarse mesh of the subdivision surface. Since the geometry is updated on the coarse level with a few control points, the geometry can be updated in a more stable manner.

For shape optimisation of lattice-skin structures, the shapes of the lattice and the thin-shell are updated at the same time, which makes the free-form deformation (FFD) method [65] an alternative for this problem. The FFD method has been applied in computational fluid dynamics for shape optimisation, for example the aerodynamic wing design [66, 67] and surface optimisation in flows [68, 69]. In the free-form deformation method, different types of geometries are embedded in the same control volume so that the updates of these geometries are entirely dependent on the deformation of the control volume. Figure 1.7 gives an example of a torus and its FFD control volume. The parametric coordinates of the torus in the control volume are computed so that



(a) A torus with control volume (b) The deformed torus with control volume

Fig. 1.7 The free-form deformation of a torus.

the torus geometry is connected with the control volume through the parametric coordinates. The control volume is deformed by updating its control points, and the deformed torus can be obtained by mapping from the parametric space to the global space based on the updated control volume as shown in Figure 1.7b.

1.3 Structure of the dissertation

Without loss of generality, a subdivision surface is used to represent the mid-surface of the thin-shell in the lattice-skin structure. Chapter 2 gives a brief review on subdivision surfaces, including subdivision schemes that are commonly used in geometry representations and the conversion to Bézier representations that will be used in the intersection computation.

Chapter 3 describes the generation of lattice-skin structures. The methods for shape interrogation of spline surfaces are first summarised and compared. Among them, the algebraic geometry approach based on implicit matrix representations will be presented in detail. A line/subdivision surface intersection algorithm is proposed afterwards, including the hierarchical bounding volume trees of k-dops for intersection detection and implicit matrix based intersection computation which solves a generalised eigenvalue problem with a few singular value decompositions.

The isogeometric analysis of lattice-skin structures is demonstrated in Chapter 4, including the analysis of subdivision thin-shells and the lattice-skin coupling method using Lagrange multipliers. Chapter 5 discusses the optimisation of lattice structures with structural compliance as the objective. The sensitivity analysis for the size and shape optimisation of lattice structures is derived in detail. In addition, a SIMP-like lattice topology optimisation is proposed for optimising the topology of lattice structures.

Chapter 6 presents lattice topology and shape optimisation of lattice-skin structures with structural compliance as the objective. The SIMP-like method proposed in Chapter 5 is applied to the lattice-skin topology optimisation. In addition, the free-form deformation technique is applied in shape optimisation of the lattice-skin structure, where the shape sensitivity analysis is given in detail with the consideration of lattice-skin coupling. Finally, Chapter 7 summarises the research work presented in this dissertation and discusses some possible future work.

Chapter 2

Review of subdivision surfaces

Subdivision surfaces are widely used in the computer graphics and animation community as a generalisation of B-splines and NURBS for modelling smooth free-form surfaces. One advantage of using subdivision surfaces compared with B-splines or NURBS which are strictly defined with structured control meshes is that a watertight smooth surface can be guaranteed with subdivision surfaces even with unstructured meshes. In constrast, B-spline and NURBS surfaces need to be trimmed and pieced together to describe a geometry with arbitrary topology and the smoothness cannot be guaranteed in the trimming and piecing together processes.

The basic idea of subdivision is that a smooth curve or surface can be generated as the limit of a sequence of successive refinements [70]. This refinement process includes two steps in general: inserting new vertices based on existing vertices and then updating the existing vertices. Different subdivision schemes are mostly derived based on spline theories, for example B-splines and box splines. In this chapter, the connection between splines and the subdivision is first demonstrated with B-splines in the univariate case. Afterwards, two different subdivision surface schemes, the Catmull-Clark subdivision scheme for quadrilateral meshes and the Loop subdivision scheme for triangular meshes, are introduced. The conversion of the subdivision to Bézier representations is given in the end, which are required in the line/subdivision surface intersection algorithm developed later.

2.1 Univariate subdivision

2.1.1 B-splines from subdivision

A B-spline basis function is a piecewise polynomial with each segment consisting of a polynomial of the same degree as the basis function. The continuity at the boundary of two adjacent segments has a higher order for B-spline basis functions of higher degrees. Therefore, for the finite element discretisation which requires a higher order of continuity in basis functions, it is straightforward to use B-spline basis functions that satisfy the continuity requirement. In addition, the B-spline basis function has a local support property, that is, the B-spline basis function associated with a control point only influences the region near the control point at some distance, which is an appealing feature for geometry modelling.

B-spline basis functions can be generated with repeated convolution starting from a box function $b^0(t)$ defined as

$$b^0(t) = \begin{cases} 1, & 0 \leq t \leq 1\,; \\ 0, & \text{otherwise}\,. \end{cases} \tag{2.1}$$

A B-spline basis function $b^\mu(t)$ of degree μ can then be obtained by calculating the convolution of the basis function $b^{\mu-1}(t)$ of degree $\mu - 1$ with the box function $b^0(t)$, that is,

$$b^\mu(t) = \left(b^{\mu-1} \otimes b^0\right)(t) = \int b^{\mu-1}(s) b^0(t-s)\,\mathrm{d}s\,. \tag{2.2}$$

The connection between B-splines and the univariate subdivision is reflected from the refinability property of B-spline basis functions [70] as follows,

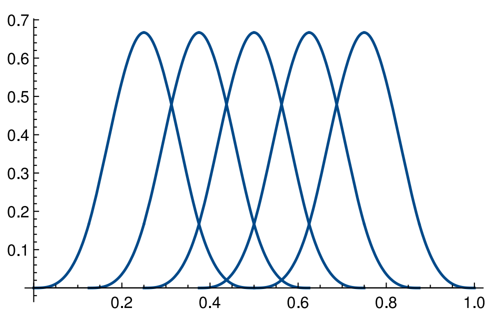
$$b^\mu(t) = \frac{1}{2^\mu} \sum_{k=0}^{\mu+1} \binom{\mu+1}{k} b^\mu(2t-k)\,. \tag{2.3}$$

It can be seen from (2.3) that a B-spline basis function can be generated by a linear combination of translated and dilated copies of itself, which is illustrated in Figure 2.1 with a cubic B-spline basis function.

The refinement equation (2.3) of B-spline basis function can be written in matrix form

$$\overline{\boldsymbol{b}}_\mu(t) = \overline{\boldsymbol{b}}_\mu(2t)\boldsymbol{S}\,, \tag{2.4}$$

where $\overline{\boldsymbol{b}}_\mu(t)$ is a row vector of B-spline basis and $\overline{\boldsymbol{b}}_\mu(2t)$ is a row vector of dilated B-spline basis, $\boldsymbol{S}$ is the refinement matrix.


(a) Translated and dilated cubic B-splines

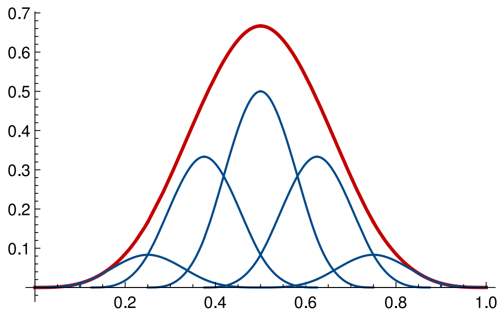

(b) Linear combination of cubic B-splines

Fig. 2.1 The refinement of a cubic B-spline basis function.

A B-spline curve $\mathcal{C}(t)$ of degree μ is defined with a set of control points $\boldsymbol{x}_i$ and associated B-spline basis functions $b_i^\mu(t)$,

$$\mathcal{C}(t) = \sum_{i=1}^{\mu+1} b_i^\mu(t)\boldsymbol{x}_i = \overline{\boldsymbol{b}}_\mu(t)\boldsymbol{x} \tag{2.5}$$

with $\boldsymbol{x}$ a column vector of control points, $\boldsymbol{x} = (\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_{\mu+1})^\mathsf{T}$.

By using the refinement equation (2.4), the same B-spline curve with control points $\boldsymbol{x}^{(0)}$ can be represented with a set of refined control points $\boldsymbol{x}^{(1)}$,

$$\mathcal{C}(t) = \overline{\boldsymbol{b}}_\mu(t)\boldsymbol{x}^{(0)} = \overline{\boldsymbol{b}}_\mu(2t)\boldsymbol{S}\boldsymbol{x}^{(0)} = \overline{\boldsymbol{b}}_\mu(2t)\boldsymbol{x}^{(1)} \tag{2.6}$$

with

$$\boldsymbol{x}^{(1)} = \boldsymbol{S}\boldsymbol{x}^{(0)} \tag{2.7}$$

defining the first-level refined control points for the B-spline curve $\mathcal{C}(t)$, and the matrix $\boldsymbol{S}$ is called the subdivision matrix in this context. The refined control points on subsequent levels can be obtained by applying (2.4) successively, and the refined control points on the subdivision level k can be computed with

$$\boldsymbol{x}^{(k)} = \boldsymbol{S}\boldsymbol{x}^{(k-1)} = \boldsymbol{S}^k\boldsymbol{x}^{(0)}\,, \tag{2.8}$$

where $\boldsymbol{S}^k$ denotes the multiplication of the subdivision matrix $\boldsymbol{S}$ for k times. Hence, the same B-spline $\mathcal{C}(t)$ defined with the refined control points on the k-th subdivision level is given as

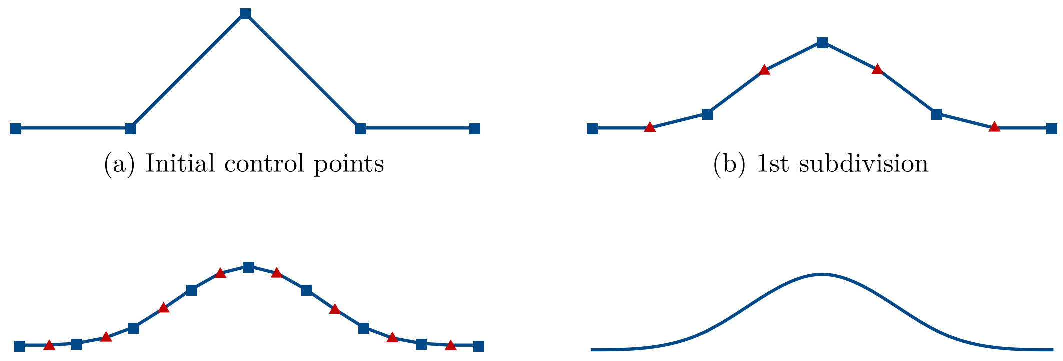
$$\mathcal{C}(t) = \overline{\boldsymbol{b}}_\mu(2^k t)\boldsymbol{x}^{(k)}\,. \tag{2.9}$$

2.1.2 Analysis of subdivision

In order to study the local properties of a subdivision curve in the neighbourhood of a point of interest, a local subdivision matrix instead of the whole subdivision matrix is needed. For example, the local subdivision matrix of a cubic B-spline is as follows,

$$\begin{bmatrix} \boldsymbol{x}_{-3}^{(k+1)} \\ \boldsymbol{x}_{-2}^{(k+1)} \\ \boldsymbol{x}_{1}^{(k+1)} \\ \boldsymbol{x}_{2}^{(k+1)} \\ \boldsymbol{x}_{3}^{(k+1)} \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 1 & 6 & 1 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 1 & 6 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{-3}^{(k)} \\ \boldsymbol{x}_{-2}^{(k)} \\ \boldsymbol{x}_{1}^{(k)} \\ \boldsymbol{x}_{2}^{(k)} \\ \boldsymbol{x}_{3}^{(k)} \end{bmatrix} . \tag{2.10}$$

The refined control points can be computed immediately from the existing control points with the local subdivision matrix. Figure 2.2 illustrates the subdivision process of generating a cubic B-spline from the initial five control points. The blue square points are those inherited from the previous subdivision level, and the red triangle points are the new ones generated at the current subdivision level. The limit curve turns out to be a cubic B-spline. In addition, the local properties of the cubic B-spline curve are always influenced only by the five control points in the neighbourhood of the position of interest.


(a) Initial control points

(b) 1st subdivision

(c) 2nd subdivision

(d) Limit curve

Fig. 2.2 Subdivision process to generate a cubic B-spline.

Some local properties of subdivision curves, for example limit positions, tangents, the convergence, etc., can be revealed from the eigendecomposition of the local asymmetric subdivision matrix $\boldsymbol{S} = \boldsymbol{S}^{\mathrm{R}} \boldsymbol{\lambda} \boldsymbol{S}^{\mathrm{L}}$. In the following context, without being clarified, the subdivision matrix $\boldsymbol{S}$ refers to the local subdivision matrix for the sake of simplicity.

For a subdivision matrix of size $n \times n$, let its right eigenvectors $\boldsymbol{S}_i^{\mathrm{R}}$ constitute the columns of the matrix $\boldsymbol{S}^{\mathrm{R}}$ and the left eigenvectors $\boldsymbol{S}_j^{\mathrm{L}}$ constitute the rows of the matrix $\boldsymbol{S}^{\mathrm{L}}$, we have

$$\boldsymbol{S}^{\mathrm{L}}\boldsymbol{S}^{\mathrm{R}} = \boldsymbol{I} \quad \text{and} \quad \boldsymbol{S}_j^{\mathrm{L}} \cdot \boldsymbol{S}_i^{\mathrm{R}} = \delta_{ij} \,. \tag{2.11}$$

The vector of initial control points $\boldsymbol{x}^{(0)} = (\boldsymbol{x}_1^{(0)} \; \boldsymbol{x}_2^{(0)} \; \cdots \; \boldsymbol{x}_n^{(0)})^{\mathsf{T}}$ can be expressed as

$$\boldsymbol{x}^{(0)} = \sum_{j=1}^{n} (\boldsymbol{S}_j^{\mathrm{L}} \cdot \boldsymbol{x}^{(0)}) \boldsymbol{S}_j^{\mathrm{R}} = \sum_{j=1}^{n} \boldsymbol{a}_j \boldsymbol{S}_j^{\mathrm{R}} \,, \tag{2.12}$$

where

$$\boldsymbol{a}_j = \boldsymbol{S}_j^{\mathrm{L}} \cdot \boldsymbol{x}^{(0)} = S_{j,1}^{\mathrm{L}} \boldsymbol{x}_1^{(0)} + S_{j,2}^{\mathrm{L}} \boldsymbol{x}_2^{(0)} + \cdots + S_{j,n}^{\mathrm{L}} \boldsymbol{x}_n^{(0)} \tag{2.13}$$

has the same dimension as a control point.

The refined control points obtained with (2.8) can be written as

$$\boldsymbol{x}^{(k)} = \sum_{j=1}^{n} \boldsymbol{a}_j \boldsymbol{S}^k \boldsymbol{S}_j^{\mathrm{R}} = \sum_{j=1}^{n} \boldsymbol{a}_j \left(\lambda_j\right)^k \boldsymbol{S}_j^{\mathrm{R}} \,, \tag{2.14}$$

where λ_j are eigenvalues of the subdivision matrix $\boldsymbol{S}$. It is indicated from (2.14) that in order to obtain the convergence of the subdivision, the largest eigenvalue λ_1 must not be greater than 1; otherwise, the refined points would have no bounds. In addition, it can be proven that the largest eigenvalue should be 1 in order to guarantee the invariance under affine transformation such as translation and rotation.

Consider a translation vector $\boldsymbol{d}$ applied to the control points $\boldsymbol{x}^{(j)}$ at the j-th subdivision level, the subdivision of the translated control points yields

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{S}(\boldsymbol{x}^{(j)} + \boldsymbol{1} \otimes \boldsymbol{d}) = \boldsymbol{S}\boldsymbol{x}^{(j)} + \boldsymbol{S}(\boldsymbol{1} \otimes \boldsymbol{d}) \,, \tag{2.15}$$

where $\boldsymbol{1}$ is a vector of size n with all ones, and $\boldsymbol{1} \otimes \boldsymbol{d}$ representing translating all the control points by $\boldsymbol{d}$. In order to guarantee the invariance under translation, the subdivision of the translated control points should be translated by the same translation vector $\boldsymbol{d}$, leading to

$$\boldsymbol{S}(\boldsymbol{1} \otimes \boldsymbol{d}) = \boldsymbol{1} \otimes \boldsymbol{d} \,, \tag{2.16}$$

indicating that the sum of entries in each row of the subdivision matrix $\boldsymbol{S}$ equals 1. Therefore, $\lambda_1 = 1$ should be an eigenvalue of $\boldsymbol{S}$ and the corresponding eigenvector is $\boldsymbol{1}$. It can also be proven that there is only one eigenvalue with magnitude 1 [70]. In other words, $\lambda_1 = 1$ and $\lambda_i < 1 (i \geq 2)$.

The limit position defined by the set of control points $\boldsymbol{x}^{(0)}$ can be obtained readily from (2.14) with eigenvalues $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_n$, that is,

$$\boldsymbol{x}^{(\infty)} = \boldsymbol{a}_1 \boldsymbol{S}_1^{\mathrm{R}} = \mathbf{1} \otimes \boldsymbol{a}_1 = \mathbf{1} \otimes (\boldsymbol{S}_1^{\mathrm{L}} \cdot \boldsymbol{x}^{(0)}) \tag{2.17}$$

with $\mathbf{1}$ a vector of size n with all ones. As can be seen, all the control points converge to the same point in the limit

$$\boldsymbol{a}_1 = \boldsymbol{S}_1^{\mathrm{L}} \cdot \boldsymbol{x}^{(0)} = S_{1,1}^{\mathrm{L}} \boldsymbol{x}_1^{(0)} + S_{1,2}^{\mathrm{L}} \boldsymbol{x}_2^{(0)} + \cdots + S_{1,n}^{\mathrm{L}} \boldsymbol{x}_n^{(0)} \, . \tag{2.18}$$

Assume that the limit position defined by the control points $\boldsymbol{x}^{(0)}$ is at the origin of the coordinate system, i.e. $\boldsymbol{a}_1 = \mathbf{0}$, we have

$$\frac{\boldsymbol{x}^{(k)}}{(\lambda_2)^k} = \boldsymbol{a}_2 \boldsymbol{S}_2^{\mathrm{R}} + \sum_{j=3}^{n} \boldsymbol{a}_j \left(\frac{\lambda_j}{\lambda_2} \right)^k \boldsymbol{S}_j^{\mathrm{R}} \, . \tag{2.19}$$

The limit of this expression gives the tangent direction at the point. If $\lambda_3 = \lambda_2$, the tangent at the point would be spanned by two vectors $\boldsymbol{a}_2$ and $\boldsymbol{a}_3$, which in general are not in the same direction. Therefore, the existence of a tangent implies that $\lambda_3 < \lambda_2$, and furthermore, all the other eigenvalues except λ_1 are smaller than λ_2. In other words, for a subdivision curve,

$$\lambda_1 = 1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_n \, . \tag{2.20}$$

leading to

$$\lim_{k \to \infty} \frac{\boldsymbol{x}^{(k)}}{(\lambda_2)^k} = \boldsymbol{a}_2 \boldsymbol{S}_2^{\mathrm{R}} \, , \tag{2.21}$$

which indicates that all the control points converge to a direction defined by $\boldsymbol{a}_2$, that is, the limit tangent defined by the control points $\boldsymbol{x}^{(0)}$ is

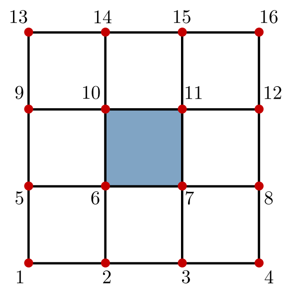
$$\boldsymbol{t} = \boldsymbol{a}_2 = \boldsymbol{S}_2^{\mathrm{L}} \cdot \boldsymbol{x}^{(0)} \, . \tag{2.22}$$

2.2 Bivariate subdivision

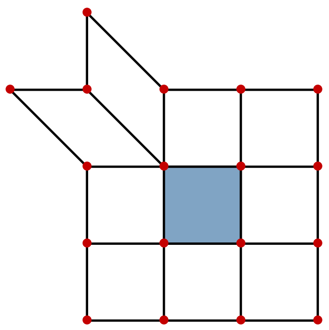
For surface meshes with arbitrary topology, several different subdivision schemes have been designed to generalise the spline theories, such as the Catmull-Clark subdivision [71], the Doo-Sabin subdivision [72] and the Loop subdivision [73]. Both the

Catmull-Clark subdivision and the Doo-Sabin subdivision were designed for quadrilateral meshes, and the Loop subdivision was derived for triangular meshes.

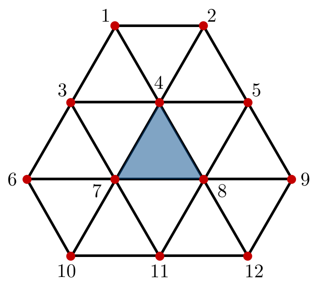
For each vertex in the mesh, a valence value representing the number of edges intersecting at the vertex is defined. For a quadrilateral mesh, if the valence of a vertex is not four, it is an extraordinary vertex (EV); for a triangular mesh, a vertex is an EV if its valence is not six. In general, all the extraordinary vertices can be isolated from each other after one subdivision step, which is useful when the subdivision surface is evaluated. The 1-ring structures are also defined for the mesh. The 1-ring of a vertex contains its neighbouring vertices connected by edges. The 1-ring of a facet contains its neighbouring facets which share at least one common vertex. The 1-ring structures of a facet in quadrilateral meshes and triangular meshes are shown in Figure 2.3.



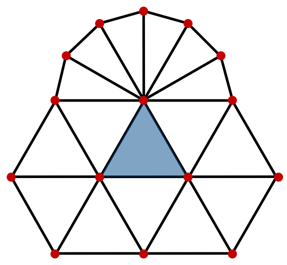
(a) A quadrilateral facet with no EVs



(b) A quadrilateral facet with an EV



(c) A triangular facet with no EVs



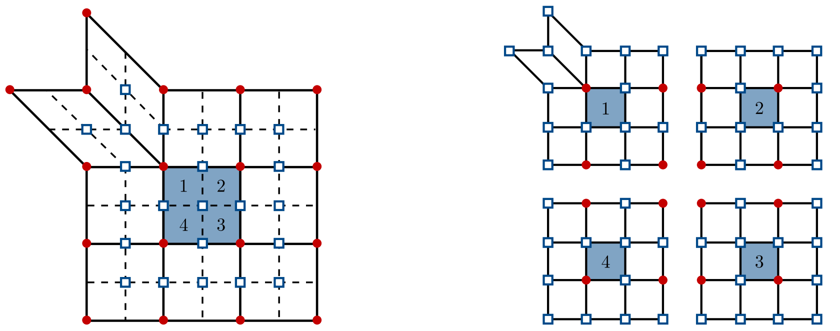
(d) A triangular facet with an EV

Fig. 2.3 One ring structures of quadrilateral and triangular facets.

Since subdivision rules are derived from splines, they inherit the local support property of splines, that is, a point on the surface is only influenced by the control points within a local support distance instead of all the control points that define the whole surface. Due to this local property, a mesh can be seen as a collection of patches defined with individual facets and their 1-rings. The patch of a facet is the only information that is required for subdividing the facet.

2.2.1 Catmull-Clark subdivision

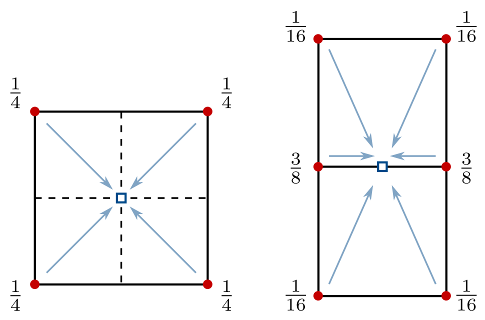
The Catmull-Clark subdivision scheme was designed to generalise the tensor-product cubic B-spline surfaces to quadrilateral meshes with arbitrary topology [71]. For the region away from an extraordinary vertex, the Catmull-Clark subdivision is equivalent to uniform cubic B-spline knot insertion. Therefore, a regular patch in the mesh containing 16 vertices, of which the vertices belonging to the centred facet are not extraordinary as shown in Figure 2.3a, defines a bi-cubic B-spline surface, and this bi-cubic B-spline surface is actually the limit surface of the patch.



(a) A Catmull-Clark patch (b) Four subpatches after one subdivision

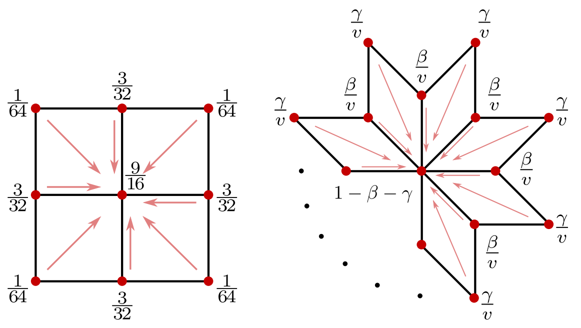
Fig. 2.4 Subdivision of a Catmull-Clark patch.

A patch in the Catmull-Clark subdivision is subdivided into four subpatches after inserting new vertices and updating existing vertices. For the patch with an EV depicted in Figure 2.4a, one of the subpatches inherits the EV with updated coordinates and other subpatches are all regular (see Figure 2.4b). The vertices in the subpatches are generated with the Catmull-Clark subdivision scheme given in Figure 2.5 and Figure 2.6, which is paraphrased based on the subdivision rules given in [71]. The new vertices within the facet and on the edges are created according to the stencils shown in Figure 2.5a and 2.5b. The existing vertices are updated in terms of their one-ring vertices and themselves with the stencils given in Figure 2.6a and 2.6b. Note that the stencil shown in Figure 2.6b becomes the stencil for a regular vertex when the valence $v = 4$. As a matter of fact, the stencils given by [71] cannot guarantee a good surface property around the extraordinary vertices. The coefficients in the stencils for



(a) Facet vertex (b) Edge vertex

Fig. 2.5 Stencils of new vertices.



(a) Regular vertex (b) Extraordinary vertex ($\beta = \frac{3}{2v}$, $\gamma = \frac{1}{4v}$ with v the valence)

Fig. 2.6 Stencils of existing vertices.

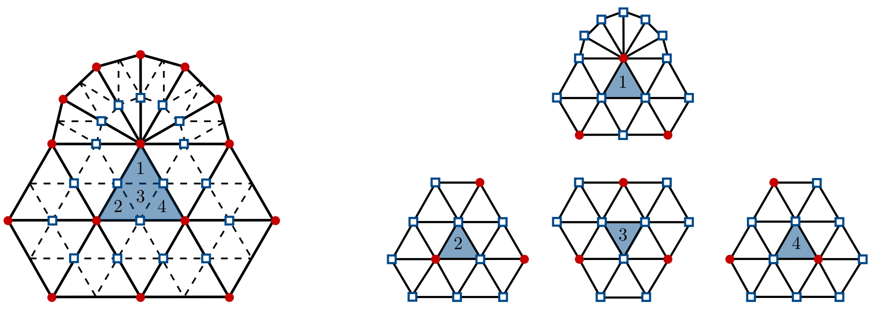
the region around EVs can be modified in order to improve the quality of the surface, for example, a significantly small Gaussian curvature variation around EVs can be obtained with the modified coefficients proposed in [17, 74].

2.2.2 Loop subdivision

The Loop subdivision scheme was designed for triangular meshes which generalises the three-direction quartic box splines [73] for the region away from extraordinary vertices. The definition of box spline and some of its properties are given in Appendix A. More details about box splines are referred to [75]. For a Loop patch with an EV as shown in Figure 2.7a, one of its subpatches after one subdivision also contains an EV, and all the other three subpatches are regular (see Figure 2.7b). The Loop patch is subdivided by inserting new vertices on edges and updating existing vertices with stencils shown in Figure 2.8. Note that the stencil for updating existing vertices given by Figure 2.8b is also applicable for regular vertices when the valence $v = 6$.

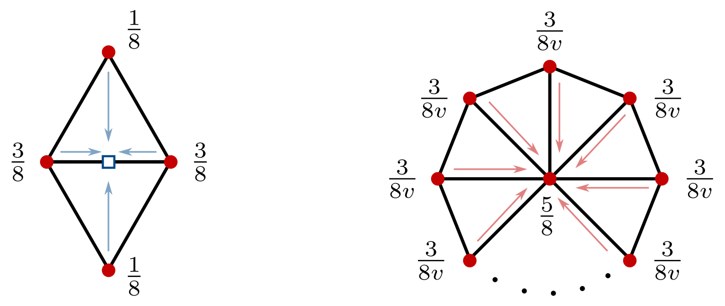
2.2.3 Extended subdivision

The traditional subdivision schemes described above for quadrilateral and triangular meshes lead to smooth surfaces. However, in the real world sharp corners and creases are common in surfaces, and the extended subdivision schemes are necessary to handle these boundaries. The extended subdivision schemes for creases and corners was proposed by Biermann et al. [76]. The crease edges, crease vertices and corners are tagged in the mesh so that the extended subdivision schemes for these different boundaries can be applied accordingly.



(a) A Loop patch (b) Four subpatches after one subdivision

Fig. 2.7 Subdivision of a Loop patch.



(a) Edge vertex (b) Existing vertex (v is the valence of the vertex)

Fig. 2.8 Loop subdivision stencils.

In the extended subdivision scheme, corner vertices are interpolated, that is, the weights in the stencil of a corner vertex are zeros for its one-ring vertices and one for itself. Therefore, the position of a corner vertex is fixed after the subdivision. For existing vertices on crease edges, the updating scheme follows the cubic B-spline interpolation. The vertex is updated with the average of its current position with the weight $\frac{3}{4}$ and the positions of its two adjacent vertices on the crease edge with weights $\frac{1}{8}$. The advantage of using the cubic B-spline interpolation is that the refinement only depends on the vertices belonging to the crease edge and is not affected by interior vertices, and it is possible to connect the surface with another subdivision surface which also supports the cubic B-spline boundary. In addition, the cubic B-spline interpolation is considered for the crease edges as it is essentially a cubic B-spline along the edge of a Catmull-Clark subdivision surface and a Loop subdivision surface.

The regular Catmull-Clark subdivision scheme in Figure 2.5a is used for new facet vertices regardless of the existence of tagged vertices and edges in the patch. For the new vertex on the crease edge, the average of the two adjacent existing crease vertices is taken as its position, which is also coincident with the cubic B-spline interpolation. The weights in the stencil of the new vertex on an untagged edge involve some treatments depending on the adjacent tagged vertices, as shown in Figure 2.9. The value of ζ is given by $\zeta = 3/8 - 1/4\cos\theta$ for quadrilateral facets and $\zeta = 1/2 - 1/4\cos\theta$ for triangular facets with $\theta = 2\pi/v$ or $\theta = \pi/v$ if the tagged vertex is on the crease, and v is the valence of the vertex [76].

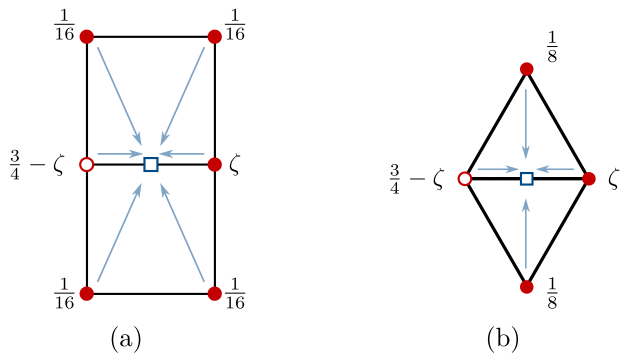
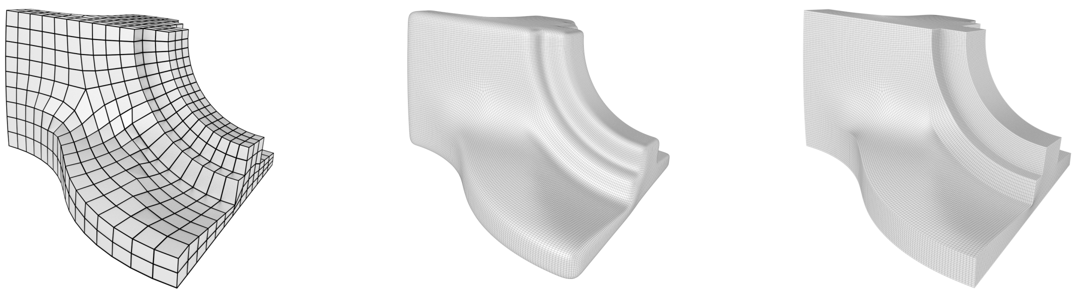


Fig. 2.9 New edge vertex in extended subdivision schemes (tagged vertices are marked with circles).

Figure 2.10 shows the difference in the limit surface of a fandisk mesh (Figure 2.10a) when using traditional Catmull-Clark subdivision scheme and the extended subdivision scheme. The traditional Catmull-Clark subdivision generates a smooth surface (Figure 2.10b), while the extended subdivision scheme results in a surface with sharp edges and corners (Figure 2.10c) as expected.



(a) Control mesh (b) Smooth surface with the Catmull-Clark subdivision (c) Sharp features with the extended subdivision

Fig. 2.10 Subdivision of a fandisk control mesh.

2.3 Conversion to Bézier representations

As described in preceding sections, the subdivision schemes commonly used are derived from spline theories. Evidently, all splines have corresponding Bézier representations. For example, a B-spline curve can be converted to a Bézier curve of the same degree, and a tensor-product B-spline surface and a three-direction quartic box spline surface can be converted to a collection of Bézier surface patches. In this section the conversions to Bézier representations from a B-spline curve, a tensor-product B-spline surface and a three-direction quartic box spline surface are introduced.

2.3.1 Bézier curves

A Bézier curve $\mathcal{C}(t)$ of degree μ is defined as

$$\mathcal{C}_B(t) = \sum_{i=1}^{\mu+1} B_i^{\mu}(t)\boldsymbol{x}_i = \overline{\boldsymbol{B}}_{\mu}(t)\boldsymbol{x}_B\,, \tag{2.23}$$

where $\boldsymbol{x}_B$ is a column vector containing control points $\boldsymbol{x}_i$, $\overline{\boldsymbol{B}}_{\mu}(t)$ is a row vector of Bernstein polynomials $B_i^{\mu}(t)$ given by

$$B_i^{\mu}(t) = \binom{\mu}{i-1} t^{i-1}(1-t)^{\mu-i+1}\,. \tag{2.24}$$

Since a B-spline curve (2.5) is a polynomial of t, the same curve can be represented with a Bézier curve of the same degree in terms of Bernstein basis functions in the form (2.23). Consider the cubic B-spline basis function as an example. It is a piecewise polynomial curve, and each polynomial segment can be represented with a cubic Bézier curve. With the end conditions of the cubic B-spline basis function as well as the C^2 requirement at the junctions of Bézier curve segments, the Bézier coefficients, as shown in Figure 2.11, can be derived for the cubic B-spline basis function.

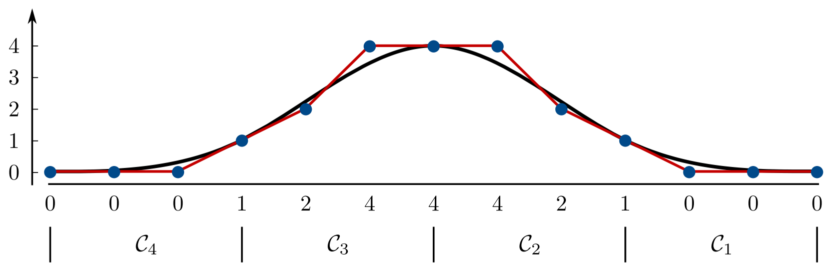


Fig. 2.11 Bézier representations of a cubic B-spline with Bézier coefficients given (the Bézier coefficients shown should be normalised by a factor of 1/6).

The four Bézier segments of the cubic B-spline basis function shown in Figure 2.11 are written explicitly as follows,

$$\mathcal{C}_1(t) = \frac{1}{6}(1-t)^3\,, \tag{2.25a}$$

$$\mathcal{C}_2(t) = \frac{4}{6}(1-t)^3 + \frac{4}{6}\cdot 3t(1-t)^2 + \frac{2}{6}\cdot 3t^2(1-t) + \frac{1}{6}t^3\,, \tag{2.25b}$$

$$\mathcal{C}_3(t) = \frac{1}{6}(1-t)^3 + \frac{2}{6}\cdot 3t(1-t)^2 + \frac{4}{6}\cdot 3t^2(1-t) + \frac{4}{6}t^3\,, \tag{2.25c}$$

$$\mathcal{C}_4(t) = \frac{1}{6}t^3\,. \tag{2.25d}$$

These polynomials are exactly the four basis functions associated with the four control points influencing the point of interest. Therefore, the cubic B-spline basis function can be written in terms of Bernstein polynomials as

$$\overline{\boldsymbol{b}}_3(t) = \overline{\boldsymbol{B}}_3(t)\boldsymbol{Q}\,, \tag{2.26}$$

where the transformation matrix $\boldsymbol{Q}$ consists of the Bézier coefficients,

$$\boldsymbol{Q} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}\,. \tag{2.27}$$

Hence, the conversion of a cubic B-spline curve to the equivalent Bézier representation is given by

$$\mathcal{C}(t) = \overline{\boldsymbol{b}}_3(t)\boldsymbol{x}_b = \overline{\boldsymbol{B}}_3(t)\boldsymbol{Q}\boldsymbol{x}_b\,. \tag{2.28}$$

Evidently, the control points $\boldsymbol{x}_B$ for the Bézier curve are

$$\boldsymbol{x}_B = \boldsymbol{Q}\boldsymbol{x}_b\,. \tag{2.29}$$

2.3.2 Tensor-product Bézier surfaces

A tensor-product Bézier surface of bi-degree (μ_1, μ_2) is defined as

$$\mathcal{S}_B(u,v) = \sum_{\boldsymbol{i}} B_{\boldsymbol{i}}^{\boldsymbol{\mu}}\boldsymbol{x}_{\boldsymbol{i}} = \sum_{i_1=1}^{\mu_1+1}\sum_{i_2=1}^{\mu_2+1} B_{i_1}^{\mu_1}(u)B_{i_2}^{\mu_2}(v)\boldsymbol{x}_{i_1 i_2}\,, \tag{2.30}$$

where $\boldsymbol{i} = \{i_1, i_2\}$ and $\boldsymbol{\mu} = \{\mu_1, \mu_2\}$ are multi-indices, $\boldsymbol{x_i}$ are control points, $B_{i_1}^{\mu_1}(u)$ and $B_{i_2}^{\mu_2}(v)$ are Bernstein polynomials as defined in (2.24).

A tensor-product B-spline basis $\{b_{j_1}^{\mu_1}(u) b_{j_2}^{\mu_2}(v)\}$ can be expressed in terms of a tensor-product Bézier basis $\{B_{i_1}^{\mu_1}(u) B_{i_2}^{\mu_2}(v)\}$ by considering the following transformation

$$\begin{aligned} b_{j_1}^{\mu_1}(u) b_{j_2}^{\mu_2}(v) &= \left(\sum_{i_1=1}^{\mu_1+1} B_{i_1}^{\mu_1}(u) Q_{i_1 j_1} \right) \cdot \left(\sum_{i_2=1}^{\mu_2+1} B_{i_2}^{\mu_2}(v) Q_{i_2 j_2} \right) \\ &= \sum_{i_1=1}^{\mu_1+1} \sum_{i_2=1}^{\mu_2+1} Q_{i_1 j_1} Q_{i_2 j_2} B_{i_1}^{\mu_1}(u) B_{i_2}^{\mu_2}(v) \,, \end{aligned} \tag{2.31}$$

which can also be written in the following form

$$\overline{\boldsymbol{b}}_{\boldsymbol{\mu}} = \overline{\boldsymbol{B}}_{\boldsymbol{\mu}} \boldsymbol{R} \tag{2.32}$$

with the fourth-order tensor $\boldsymbol{R}$ given by

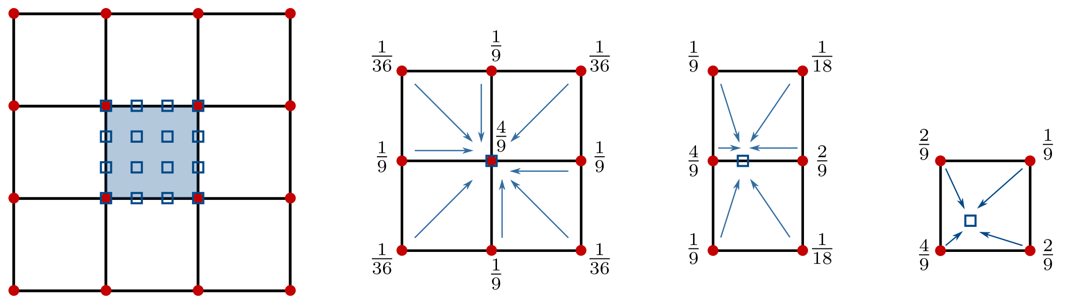
$$R_{i_1 i_2 j_1 j_2} = Q_{i_1 j_1} Q_{i_2 j_2} \,. \tag{2.33}$$

Therefore, the conversion of a tensor-product B-spline surface into the corresponding tensor-product Bézier surface is achieved with

$$\mathcal{S}_B(u,v) = \overline{\boldsymbol{b}}_{\boldsymbol{\mu}}(u,v) \boldsymbol{x}_b = \overline{\boldsymbol{B}}_{\boldsymbol{\mu}}(u,v) \boldsymbol{R} \boldsymbol{x}_b \,, \tag{2.34}$$

and the control points for the tensor-product Bézier surface are given by

$$\boldsymbol{x}_B = \boldsymbol{R} \boldsymbol{x}_b \,. \tag{2.35}$$



(a) Control points (b) Corner vertex (c) Edge vertex (d) Facet vertex

Fig. 2.12 Stencils to convert a bi-cubic B-spline surface to Bézier representation.

Specifically, for a bi-cubic B-spline surface to which the Catmull-Clark subdivision scheme generalises, the conversion to the corresponding bi-cubic Bézier surface can be achieved with transformation stencils, as shown in Figure 2.12 where the red dots denote B-spline surface control points and the blue squares denote Bézier surface control points. The Bézier control points can be obtained directly by the convolution of the B-spline control points with the stencils.

2.3.3 Triangular Bézier surfaces

A triangular Bézier surface of degree μ is defined as

$$\mathcal{S}_B(u,v,w) = \sum_{\boldsymbol{i}} B^{\mu}_{\boldsymbol{i}} \boldsymbol{x}_{\boldsymbol{i}} = \sum_{i_1+i_2+i_3=\mu+3} B^{\mu}_{i_1 i_2 i_3}(u,v,w)\boldsymbol{x}_{i_1 i_2 i_3}\,, \tag{2.36}$$

where (u,v,w) are the barycentric coordinates of a point in the triangle, $\boldsymbol{i} = \{i_1, i_2, i_3\}$ is a multi-index, and the basis function $B^{\mu}_{i_1 i_2 i_3}$ is given by

$$B^{\mu}_{i_1 i_2 i_3} = \frac{\mu!}{(i_1-1)!(i_2-1)!(i_3-1)!} u^{i_1-1} v^{i_2-1} w^{i_3-1} \tag{2.37}$$

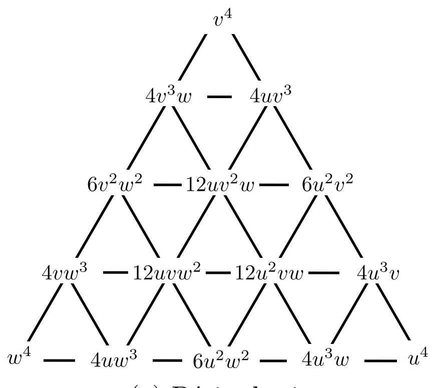
with $i_1, i_2, i_3 = 1, 2, \cdots, \mu+1$ and $u+v+w=1$.

In the Loop subdivision scheme, the surface represented by a regular Loop patch is equivalent to a three-direction quartic box spline surface. The corresponding quartic box spline basis functions can be derived after transforming the box spline patch into a triangular Bézier patch. For a three-direction quartic box spline, the evaluation of the spline itself at the local support boundary is zero, and its first and second derivatives also vanish at the local support boundary. In addition, the first and second derivatives across the triangle edges should be continuous, i.e. the box spline is C^2 across the triangle edges. Based on these conditions, the Bézier coefficients for a box spline can be obtained as in [77].

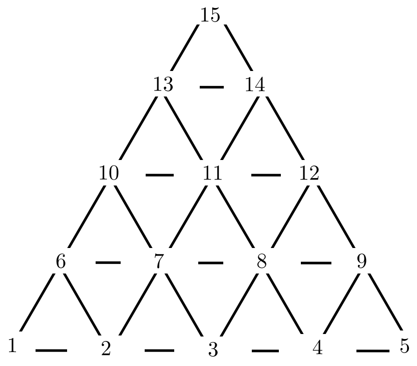
For the quartic triangular Bézier basis arranged as in Figure 2.13, the transformation matrix $\boldsymbol{R}$ is listed in Appendix A.3. With the transformation matrix, the quartic box spline basis functions can be expressed in terms of the corresponding 15 triangular Bézier basis functions, that is,

$$\overline{\boldsymbol{b}}_4(u,v,w) = \overline{\boldsymbol{B}}_4(u,v,w)\boldsymbol{R} \tag{2.38}$$

with $\overline{\boldsymbol{B}}_4(u,v,w)$ denoting a row vector of 15 Bézier basis functions.



(a) Bézier basis



(b) Indices of Bézier basis

Fig. 2.13 Quartic triangular Bézier basis functions and their indices.

Therefore, the conversion of a quartic box spline surface into a triangular Bézier surface is given by

$$\mathcal{S}_B(u,v,w) = \overline{\boldsymbol{b}}_4(u,v,w)\boldsymbol{x}_b = \overline{\boldsymbol{B}}_4(u,v,w)\boldsymbol{R}\boldsymbol{x}_b\,, \tag{2.39}$$

and the Bézier control points are computed with

$$\boldsymbol{x}_B = \boldsymbol{R}\boldsymbol{x}_b\,. \tag{2.40}$$

Chapter 3

Lattice-skin geometry generation

The lattice-skin structure contains the thin-shell and the lattice, in which the thin-shell is represented with a subdivision surface and the lattice is represented with a set of line segments. In order to generate the geometry of the lattice-skin structure, a line/subdivision surface intersection algorithm is needed. In this chapter, a new line/subdivision surface intersection algorithm is developed.

The information of the surface is required in order to compute the intersection between a line and a surface. In the context of the surface intersection, the approaches for spline surface interrogation are first reviewed in Section 3.1, including the divide-and-conquer approach, the marching approach and the implicitisation approach. The implicit matrix representation (implicit M-Rep) belonging to the implicitisation approach is adopted in the intersection algorithm.

A detailed description of the new line/subdivision surface intersection algorithm is in Section 3.2, including the intersection detection using bounding volume trees and the intersection computation with implicit M-Reps. The internal lattice generation process is described in Section 3.3, and the Stanford bunny model is demonstrated as an example in the end.

3.1 Review of surface interrogation approaches

Surface interrogation is a process to extract information about a surface, and the information obtained can be used to solve the surface-related problems [78], for example, surface intersection, offset surface, geodesics, etc., which are important applications in computer-aided engineering design. We consider only intersection problems of parametric surfaces and curves with Bézier representations. The prevalent techniques for the surface interrogation include the triangulation, the marching method, the subdivision method, the implicitisation and a combination of them. Since in the triangulation method the parametric surface is approximated with linear triangles and the information of the original surface is not remained for the intersection, it will not be discussed in this chapter. The techniques described in the following are also available to subdivision surfaces since a subdivision surface can be seen as a piecewise spline surface.

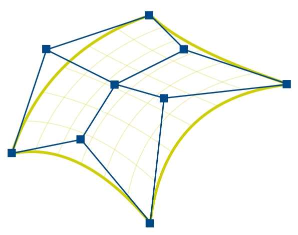
3.1.1 Divide-and-conquer

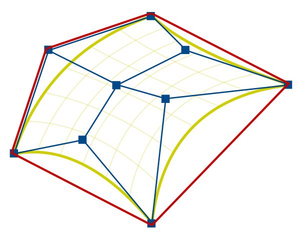
The idea of the divide-and-conquer method (also known as the subdivision method) is that the intersection problem is repeatedly divided into smaller and simpler subproblems, and the intersection is detected and computed for the subproblems. The process of the divide-and-conquer approach involves a sequence of intersection detections and surface refinements until some termination criterion is satisfied, and the intersection curve is finally approximated by connecting the intersection points in an oriented order. A detailed description of the divide-and-conquer method is as follows.

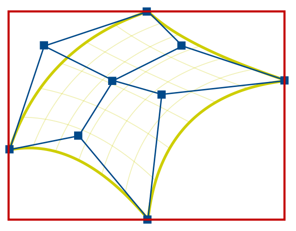
Intersection detection

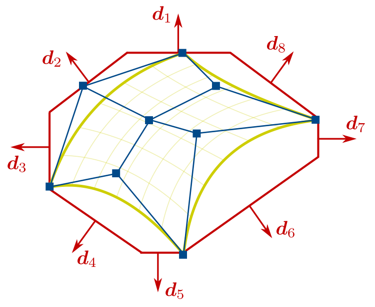
For Bézier and B-spline representations, the surface lies entirely within the convex hull of the control points due to their non-negative basis functions and the partition of unity property. The potential intersection can be detected with the convex hull of the surface. If the convex hulls of two objects (surfaces or curves) do not intersect, there is no intersection between the two objects; otherwise, there is a possibility that the two objects intersect. However, it is in general not cheap to compute the convex hull of a free-form parametric surface. Instead, the bounding box and the k-dop (a discrete orientation polytope [79] or also known as a quantised hull [80]) can be used to detect the intersection. Though the bounding box and the k-dop are not as tight as the convex hull, they are easier to construct and also effective in detecting possible intersections.

K-dops can be understood as a generalisation of axis-aligned bounding boxes so that the bounding volume is determined by a fixed k ($k \geq 6$ in 3D) directions. The axis-aligned bounding box can be seen as a 6-dop which is determined by the three coordinate axes. Compared with the bounding box, a k-dop with $k > 6$ can provide a tighter bound for the object studied and it is straightforward to compute. Figure 3.1 shows a convex hull, a bounding box and an 8-dop for a planar quadratic Bézier surface.


(a) Quadratic Bézier patch


(b) Convex hull of vertices.


(c) Bounding box of vertices



(d) An 8-dop of vertices

Fig. 3.1 Bézier patches lie entirely within the convex hull, the bounding box and the k-dop formed by their vertices.

To determine whether the k-dops of a patch and a line intersect it is not necessary to explicitly construct their k-dops. It is sufficient to consider their control points and their projections along the k prescribed directions $\boldsymbol{d}_j$, see Figure 3.2. As an example, the intersection of two k-dops belonging to a cubic Bézier curve and a line segment is illustrated in Figure 3.2. The support heights $h^{\text{c}}_{j,\max}$ and $h^{\text{c}}_{j,\min}$ of the cubic Bézier curve are defined as follows

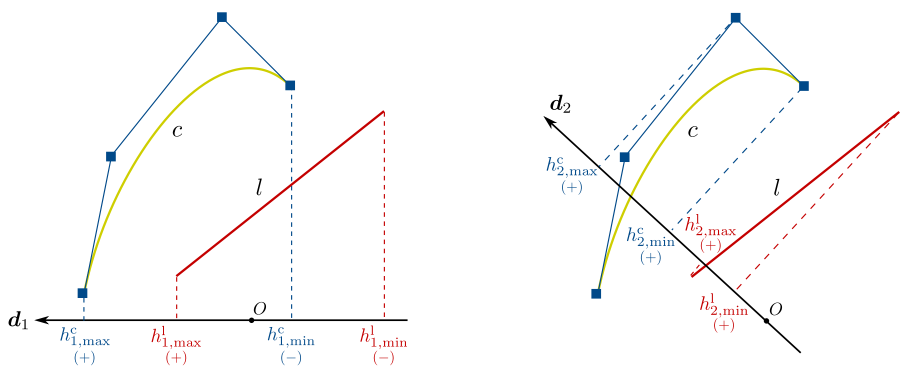
$$h^{\text{c}}_{j,\max} = \max_{\boldsymbol{i}}(\boldsymbol{x_i} \cdot \boldsymbol{d}_j) \quad \text{and} \quad h^{\text{c}}_{j,\min} = \min_{\boldsymbol{i}}(\boldsymbol{x_i} \cdot \boldsymbol{d}_j)\,, \tag{3.1}$$

where $\boldsymbol{x_i}$ are control points of the subdivision patch. The support heights $h^{\text{l}}_{j,\max}$ and $h^{\text{l}}_{j,\min}$ of the line are computed in a similar way. The two k-dops intersect, that is, potential intersection exists, only when

$$\forall\, j\,, \qquad h^{\text{c}}_{j,\max} \geq h^{\text{l}}_{j,\min} \quad \text{and} \quad h^{\text{l}}_{j,\max} \geq h^{\text{c}}_{j,\min}\,. \tag{3.2}$$

Alternatively, there is no intersection between the two k-dops if

$$\exists\, j, \quad \text{such that} \quad h^{\text{c}}_{j,\max} < h^{\text{l}}_{j,\min} \quad \text{or} \quad h^{\text{l}}_{j,\max} < h^{\text{c}}_{j,\min}\,. \tag{3.3}$$



(a) Support heights in $\boldsymbol{d}_1$

(b) Support heights in $\boldsymbol{d}_2$

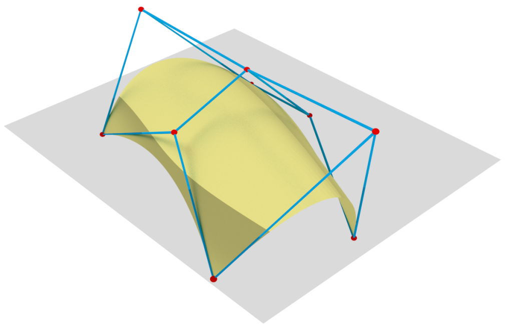
Fig. 3.2 Intersection detection between a line segment and a cubic Bézier curve. There is no intersection in $\boldsymbol{d}_2$ direction.

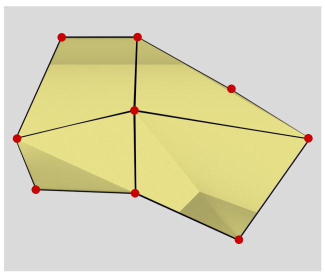
Surface refinement

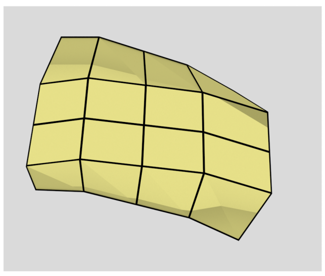
If a potential intersection is detected between two surfaces or curves, they are split into subpieces. The intersection detection is performed again to these subpieces. This process is repeated until the final small subpieces are simple (i.e. flat) enough to compute the intersection. The refinement process to split the surface or curve is straightforward for Bézier and B-spline representations [81]. De Casteljau's algorithm is available for Bézier representations, and knot refinement techniques can be applied for B-spline representations [81]. Figure 3.3 shows the refinement process in the divide-and-conquer method to obtain the intersection curve between a quadratic Bézier surface and a plane.

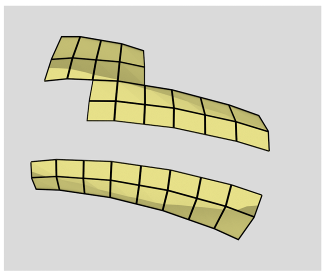
Simplicity criteria of surfaces

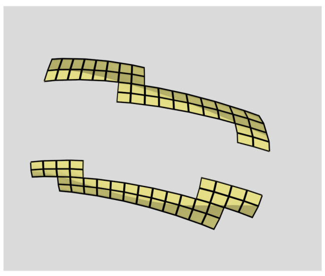
Eventually, a bounding volume tree of potentially intersecting subpieces of each surface or curve is generated. Each node in the tree contains the intersection information. When the subpieces are simple enough to be approximated by planar facets or line segments, the intersection is computed between pairs of planar facets or line segments, which becomes simple and straightforward. For the intersection between pairs of planar facets, each of the intersecting quadrilateral facets is triangulated into two triangles so that the intersections between triangles are computed. The simplicity criteria that are used to terminate the refinement process include the surface flatness and the edge linearity [82]. For a quadrilateral surface subpiece, the average normal of the control polygon of the surface subpiece is computed as the average of the normals of all the

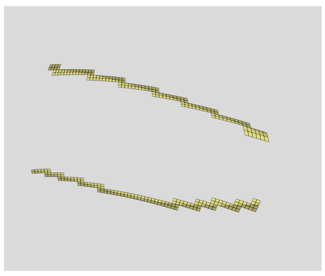

(a) Intersection between a quadratic Bézier surface and a plane


(b) Initial control polygon


(c) 1st refinement


(d) 2nd refinement


(e) 3rd refinement


(f) 5th refinement

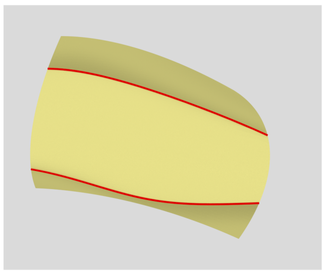
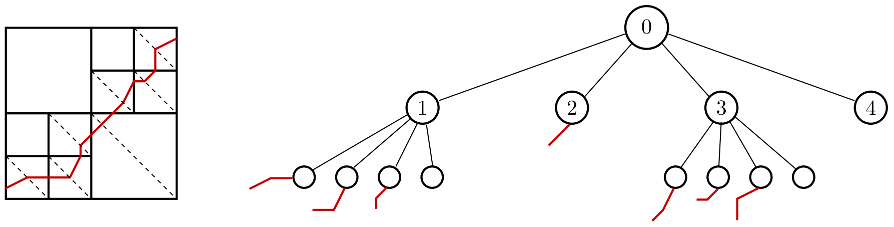

(g) Final intersection curve

Fig. 3.3 The intersection of a quadratic Bézier surface and a plane using the divide-and-conquer method.

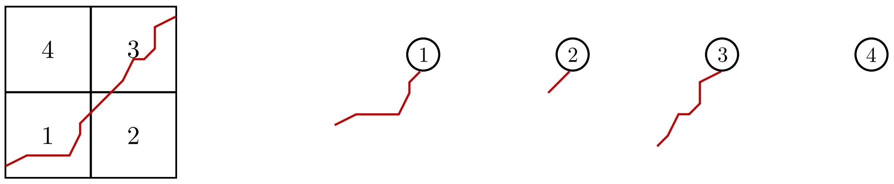
quadrilateral polygons, and the support heights $h_{j,\max}$ and $h_{j,\min}$ of the surface subpiece along the average normal are computed. The difference between these two support heights is considered as the measurement of flatness. A sufficiently small tolerance is set to the flatness measurement such that the accuracy of the intersection computation can be controlled by this tolerance. However, a flat surface subpiece may need further refinement if the edges of the surface subpiece are not approximately linear, for example planar Bézier surfaces shown in Figure 3.1. The edge linearity is controlled by the measurement of maximum angle variation along each edge of the surface subpiece. The example given in Figure 3.3 uses a tolerance of 10^{-4} and the tolerance is satisfied after nine refinement steps. In practice, if a surface subpiece is detected as intersected and the simplicity criteria are satisfied, the refinement of the surface subpiece terminates and the intersection is computed for this subpiece. As a result, the refinement depths of the final intersecting surface subpieces may not be the same as some subpieces become simple at earlier stages.

Concatenation of intersection points

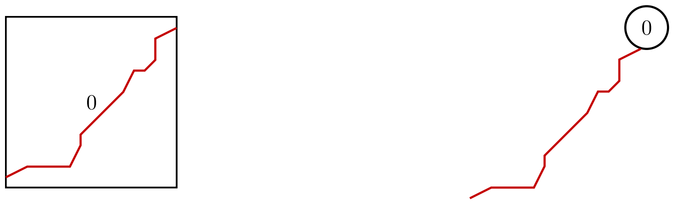
In surface/surface intersection problems, an intersection curve is approximated with a sequence of line segments, and all the line segments are connected in an end-to-end manner, that is, the finishing point of a line segment is the beginning point of the next segment. However, the intersection points obtained in the divide-and-conquer are not automatically sorted as expected. A bottom-top joining scheme was implemented in order to obtain an oriented intersection curve. At the bottom level of the divide-and-conquer process, the line segments in each patch are connected in a way such that the curve segment in that patch consisting of these line segments is oriented. The set of curve segments at the bottom level are then further joined together at the next one level up to generate a set of fewer but longer curve segments. This process is repeated until the top level is reached when all the intersection points are concatenated in an oriented order. Figure 3.4 illustrates the joining process described above. The refinement depths of the leaves in the tree shown in Figure 3.4 are considered to be different deliberately in order to demonstrate that the refinement process for some facets can stop earlier when they satisfy the simplicity criteria. The dashed lines in Figure 3.4a denote the triangulation of the intersecting planar quadrilateral facets for the plane/plane intersection computation.



(a) Oriented curve segments on the bottom level



(b) Joining curve segments from the lower level



(c) Final curve segments joined to form the intersection curve

Fig. 3.4 Joining line segments to form the final intersection curve in an end-to-end manner.

3.1.2 Marching

For the surface/surface intersection problem, the marching method is to trace an intersection curve branch by generating a sequence of intersection points from some starting points. The intersection points on an intersection curve are generated by stepping from a given starting intersection point until it reaches a finishing point or a boundary edge. In general, the marching method includes two major stages: determine a starting point on the intersection curve in the first place, followed by several marching steps with each consisting of a major step and minor steps to find the next intersection point. The stepping process is performed in the parametric spaces of the two surfaces and requires the local differential geometry information of the surfaces.

Differential geometry

Consider two parametric surfaces $\mathcal{S}_\mathrm{A}$ and $\mathcal{S}_\mathrm{B}$ defined in parameter spaces (r, s) and (u, v) respectively, points on the two surfaces are represented as $\boldsymbol{P}_\mathrm{A}(r, s)$ on $\mathcal{S}_\mathrm{A}$ and

$\boldsymbol{P}_{\mathrm{B}}(u,v)$ on $\mathcal{S}_{\mathrm{B}}$. The tangents at $\boldsymbol{P}_{\mathrm{A}}(r,s)$ and $\boldsymbol{P}_{\mathrm{B}}(u,v)$ are given by

$$\boldsymbol{P}_{\mathrm{A},r} = \frac{\partial \boldsymbol{P}_{\mathrm{A}}}{\partial r}, \quad \boldsymbol{P}_{\mathrm{A},s} = \frac{\partial \boldsymbol{P}_{\mathrm{A}}}{\partial s}\,, \tag{3.4a}$$

$$\boldsymbol{P}_{\mathrm{B},u} = \frac{\partial \boldsymbol{P}_{\mathrm{B}}}{\partial u}, \quad \boldsymbol{P}_{\mathrm{B},v} = \frac{\partial \boldsymbol{P}_{\mathrm{B}}}{\partial v}\,. \tag{3.4b}$$

The normals at $\boldsymbol{P}_{\mathrm{A}}(r,s)$ and $\boldsymbol{P}_{\mathrm{B}}(u,v)$ are defined as

$$\boldsymbol{n}_{\mathrm{A}} = \boldsymbol{P}_{\mathrm{A},r} \times \boldsymbol{P}_{\mathrm{A},s}\,, \tag{3.5a}$$

$$\boldsymbol{n}_{\mathrm{B}} = \boldsymbol{P}_{\mathrm{B},u} \times \boldsymbol{P}_{\mathrm{B},v}\,. \tag{3.5b}$$

Let $\mathcal{C}(t)$ be the intersection curve of the two surfaces, a point $\boldsymbol{P}(t_0)$ on the curve can also be evaluated with either $\boldsymbol{P}_{\mathrm{A}}(r_0,s_0)$ or $\boldsymbol{P}_{\mathrm{B}}(u_0,v_0)$. The unit tangent of the intersection curve $\mathcal{C}(t)$ at the point $\boldsymbol{P}$ is used to determine the marching direction and indicate the starting point and the finishing point. The unit tangent is computed with

$$\boldsymbol{t} = \frac{\boldsymbol{n}_{\mathrm{A}} \times \boldsymbol{n}_{\mathrm{B}}}{|\boldsymbol{n}_{\mathrm{A}} \times \boldsymbol{n}_{\mathrm{B}}|}\,. \tag{3.6}$$

Determination of starting points

A starting point $\boldsymbol{P}_0$ to invoke the marching process can be found by searching along the edges of the two surfaces. The intersection point of an edge of a surface with the other surface can be taken as the starting point, and a curve/surface intersection problem needs to be solved to find this starting point.

Consider the four edges of the surface $\mathcal{S}_{\mathrm{A}}$, $\boldsymbol{e}^1_{\mathrm{A}}(t) = \boldsymbol{P}_{\mathrm{A}}(u,0)$, $\boldsymbol{e}^2_{\mathrm{A}}(t) = \boldsymbol{P}_{\mathrm{A}}(1,v)$, $\boldsymbol{e}^3_{\mathrm{A}}(t) = \boldsymbol{P}_{\mathrm{A}}(u,1)$ and $\boldsymbol{e}^4_{\mathrm{A}}(t) = \boldsymbol{P}_{\mathrm{A}}(0,v)$, the four edges $\boldsymbol{e}^i_{\mathrm{A}}(t)$ are examined one by one to find their intersection points with the surface $\mathcal{S}_{\mathrm{B}}$. This intersection problem can be described as finding parameters t, u and v such that for an edge $\boldsymbol{e}^i_{\mathrm{A}}$,

$$\boldsymbol{e}^i_{\mathrm{A}}(t) - \boldsymbol{P}_{\mathrm{B}}(u,v) = 0\,. \tag{3.7}$$

Given initial parameters $t^{(0)}$, $u^{(0)}$ and $v^{(0)}$, the Newton-Raphson iterative approach can be used to get a converged solution. The linear approximation of the above equation gives

$$\boldsymbol{e}^i_{\mathrm{A}}(t^{(0)}) + \frac{\partial \boldsymbol{e}^i_{\mathrm{A}}}{\partial t}(t^{(0)})\delta t - \boldsymbol{P}_{\mathrm{B}}(u^{(0)},v^{(0)}) - \frac{\partial \boldsymbol{P}_{\mathrm{B}}}{\partial u}(u^{(0)},v^{(0)})\delta u - \frac{\partial \boldsymbol{P}_{\mathrm{B}}}{\partial v}(u^{(0)},v^{(0)})\delta v = 0\,, \tag{3.8}$$

which can be written in matrix form as

$$\begin{pmatrix} \boldsymbol{e}^{i(0)}_{\mathrm{A},t} & -\boldsymbol{P}^{(0)}_{\mathrm{B},u} & -\boldsymbol{P}^{(0)}_{\mathrm{B},v} \end{pmatrix} \begin{pmatrix} \delta t \\ \delta u \\ \delta v \end{pmatrix} = \begin{pmatrix} -\boldsymbol{e}^{i(0)}_{\mathrm{A},t} + \boldsymbol{P}^{(0)}_{\mathrm{B}} \end{pmatrix} . \tag{3.9}$$

The parameters are updated with

$$t^{(1)} = t^{(0)} + \delta t, \quad u^{(1)} = u^{(0)} + \delta u, \quad v^{(1)} = v^{(0)} + \delta v \, , \tag{3.10}$$

in the next iteration. The iteration proceeds until some termination condition is satisfied, and the starting intersection point $\boldsymbol{P}_0$ is selected from one of those curve/surface intersection points.

Marching steps of tracing a curve

After the starting intersection point $\boldsymbol{P}_0$ is obtained, the marching steps are performed in the parametric spaces of the two surfaces to get other points on the intersection curve. The starting point on the two surfaces are represented with $\boldsymbol{P}^{(0)}_{\mathrm{A}}(r_0, s_0)$ and $\boldsymbol{P}^{(0)}_{\mathrm{B}}(u_0, v_0)$ respectively, and $\boldsymbol{P}^{(0)}_{\mathrm{A}}(r_0, s_0) \approx \boldsymbol{P}^{(0)}_{\mathrm{B}}(u_0, v_0)$.

Let $\{\delta r, \delta s\}$ and $\{\delta u, \delta v\}$ be the marching step in the neighbourhood of $\boldsymbol{P}^{(0)}_{\mathrm{A}}$ and $\boldsymbol{P}^{(0)}_{\mathrm{B}}$ in the parametric spaces, the linear approximations of the next iterative points $\boldsymbol{P}^{(1)}_{\mathrm{A}}$ and $\boldsymbol{P}^{(1)}_{\mathrm{B}}$ are

$$\boldsymbol{P}^{(1)}_{\mathrm{A}}(r_1, s_1) = \boldsymbol{P}_{\mathrm{A}}(r_0 + \delta r, s_0 + \delta s) \approx \boldsymbol{P}_{\mathrm{A}}(r_0, s_0) + \boldsymbol{P}_{\mathrm{A},r}(r_0, s_0)\delta r + \boldsymbol{P}_{\mathrm{A},s}(r_0, s_0)\delta s \, , \tag{3.11a}$$

$$\boldsymbol{P}^{(1)}_{\mathrm{B}}(u_1, v_1) = \boldsymbol{P}_{\mathrm{B}}(u_0 + \delta u, v_0 + \delta v) \approx \boldsymbol{P}_{\mathrm{B}}(u_0, v_0) + \boldsymbol{P}_{\mathrm{B},u}(u_0, v_0)\delta u + \boldsymbol{P}_{\mathrm{B},v}(u_0, v_0)\delta v \, . \tag{3.11b}$$

The new point $\boldsymbol{P}^{(1)}_{\mathrm{A}}$ needs to lie on the tangent plane at $\boldsymbol{P}^{(0)}_{\mathrm{B}}$, and the projection of $(\boldsymbol{P}^{(1)}_{\mathrm{A}} - \boldsymbol{P}_0)$ on the unit tangent $\boldsymbol{t}_0$ evaluated at $\boldsymbol{P}_0$ should be the step length δL. These two conditions yield the following equations,

$$(\boldsymbol{P}^{(1)}_{\mathrm{A}} - \boldsymbol{P}^{(0)}_{\mathrm{B}}) \cdot \boldsymbol{n}_{\mathrm{B}}(u_0, v_0) = 0 \, , \tag{3.12a}$$

$$(\boldsymbol{P}^{(1)}_{\mathrm{A}} - \boldsymbol{P}_0) \cdot \boldsymbol{t}_0 = \delta L \, . \tag{3.12b}$$

Substituting the linear approximation of $\boldsymbol{P}_\mathrm{A}^{(1)}$, the above two equations can be written in matrix form

$$\begin{pmatrix} \boldsymbol{P}_{\mathrm{A},r}^{(0)} \cdot \boldsymbol{n}_\mathrm{B}^{(0)} & \boldsymbol{P}_{\mathrm{A},s}^{(0)} \cdot \boldsymbol{n}_\mathrm{B}^{(0)} \\ \\ \boldsymbol{P}_{\mathrm{A},r}^{(0)} \cdot \boldsymbol{t}_0 & \boldsymbol{P}_{\mathrm{A},s}^{(0)} \cdot \boldsymbol{t}_0 \end{pmatrix} \begin{pmatrix} \delta r \\ \\ \delta s \end{pmatrix} = \begin{pmatrix} (\boldsymbol{P}_\mathrm{B}^{(0)} - \boldsymbol{P}_\mathrm{A}^{(0)}) \cdot \boldsymbol{n}_\mathrm{B}^{(0)} \\ \\ (\boldsymbol{P}_0 - \boldsymbol{P}_\mathrm{A}^{(0)}) \cdot \boldsymbol{t}_0 + \delta L \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \delta L \end{pmatrix} . \tag{3.13}$$

Similarly, the equations for the new point $\boldsymbol{P}_\mathrm{B}^{(1)}$ can be obtained as follows,

$$\begin{pmatrix} \boldsymbol{P}_{\mathrm{B},u}^{(0)} \cdot \boldsymbol{n}_\mathrm{A}^{(0)} & \boldsymbol{P}_{\mathrm{B},v}^{(0)} \cdot \boldsymbol{n}_\mathrm{A}^{(0)} \\ \\ \boldsymbol{P}_{\mathrm{B},u}^{(0)} \cdot \boldsymbol{t}_0 & \boldsymbol{P}_{\mathrm{B},v}^{(0)} \cdot \boldsymbol{t}_0 \end{pmatrix} \begin{pmatrix} \delta u \\ \\ \delta v \end{pmatrix} = \begin{pmatrix} (\boldsymbol{P}_\mathrm{A}^{(0)} - \boldsymbol{P}_\mathrm{B}^{(0)}) \cdot \boldsymbol{n}_\mathrm{A}^{(0)} \\ \\ (\boldsymbol{P}_0 - \boldsymbol{P}_\mathrm{B}^{(0)}) \cdot \boldsymbol{t}_0 + \delta L \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \delta L \end{pmatrix} . \tag{3.14}$$

The parameters in the parametric spaces of the two surfaces are then updated with

$$r_1 = r_0 + \delta r, \quad s_1 = s_0 + \delta s \,, \tag{3.15a}$$

$$u_1 = u_0 + \delta u, \quad v_1 = v_0 + \delta v \,. \tag{3.15b}$$

After the first marching step, the points on the two surfaces $\mathcal{S}_\mathrm{A}$ and $\mathcal{S}_\mathrm{B}$ are updated to $\boldsymbol{P}_\mathrm{A}^{(1)}(r_1, s_1)$ and $\boldsymbol{P}_\mathrm{B}^{(1)}(u_1, v_1)$, which are not the same point in general. Therefore, subsequent minor steps are performed to correct the new points obtained after the first marching step. The new points $\boldsymbol{P}_\mathrm{A}^{(k)}$ and $\boldsymbol{P}_\mathrm{B}^{(k)}$ in the minor steps can be obtained with a similar process as in the first step. Assume that the step length δL keeps the same, we have

$$(\boldsymbol{P}_\mathrm{A}^{(k)} - \boldsymbol{P}_\mathrm{B}^{(k-1)}) \cdot \boldsymbol{n}_\mathrm{B}(u_{k-1}, v_{k-1}) = 0 \,, \tag{3.16a}$$

$$(\boldsymbol{P}_\mathrm{A}^{(k)} - \boldsymbol{P}_0) \cdot \boldsymbol{t}_0 = \delta L \,, \tag{3.16b}$$

$$(\boldsymbol{P}_\mathrm{B}^{(k)} - \boldsymbol{P}_\mathrm{A}^{(k-1)}) \cdot \boldsymbol{n}_\mathrm{A}(r_{k-1}, s_{k-1}) = 0 \,, \tag{3.16c}$$

$$(\boldsymbol{P}_\mathrm{B}^{(k)} - \boldsymbol{P}_0) \cdot \boldsymbol{t}_0 = \delta L \,. \tag{3.16d}$$

The matrix forms of the above equations are given by

$$\begin{pmatrix} \boldsymbol{P}_{\mathrm{A},r}^{(k-1)} \cdot \boldsymbol{n}_\mathrm{B}^{(k-1)} & \boldsymbol{P}_{\mathrm{A},s}^{(k-1)} \cdot \boldsymbol{n}_\mathrm{B}^{(k-1)} \\ \\ \boldsymbol{P}_{\mathrm{A},r}^{(k-1)} \cdot \boldsymbol{t}_0 & \boldsymbol{P}_{\mathrm{A},s}^{(k-1)} \cdot \boldsymbol{t}_0 \end{pmatrix} \begin{pmatrix} \delta r \\ \\ \delta s \end{pmatrix} = \begin{pmatrix} (\boldsymbol{P}_\mathrm{B}^{(k-1)} - \boldsymbol{P}_\mathrm{A}^{(k-1)}) \cdot \boldsymbol{n}_\mathrm{B}^{(k-1)} \\ \\ (\boldsymbol{P}_0 - \boldsymbol{P}_\mathrm{A}^{(k-1)}) \cdot \boldsymbol{t}_0 + \delta L \end{pmatrix} , \tag{3.17}$$

and

$$\begin{pmatrix} \boldsymbol{P}_{\mathrm{B},u}^{(k-1)} \cdot \boldsymbol{n}_{\mathrm{A}}^{(k-1)} & \boldsymbol{P}_{\mathrm{B},v}^{(k-1)} \cdot \boldsymbol{n}_{\mathrm{A}}^{(k-1)} \\ \\ \boldsymbol{P}_{\mathrm{B},u}^{(k-1)} \cdot \boldsymbol{t}_0 & \boldsymbol{P}_{\mathrm{B},v}^{(k-1)} \cdot \boldsymbol{t}_0 \end{pmatrix} \begin{pmatrix} \delta u \\ \\ \delta v \end{pmatrix} = \begin{pmatrix} (\boldsymbol{P}_{\mathrm{A}}^{(k-1)} - \boldsymbol{P}_{\mathrm{B}}^{(k-1)}) \cdot \boldsymbol{n}_{\mathrm{A}}^{(k-1)} \\ \\ (\boldsymbol{P}_0 - \boldsymbol{P}_{\mathrm{B}}^{(k-1)}) \cdot \boldsymbol{t}_0 + L \end{pmatrix} . \tag{3.18}$$

The parameters of the new points $\boldsymbol{P}_{\mathrm{A}}^{(k)}$ and $\boldsymbol{P}_{\mathrm{B}}^{(k)}$ are

$$r_k = r_{k-1} + \delta r, \quad s_k = s_{k-1} + \delta s\,, \tag{3.19a}$$

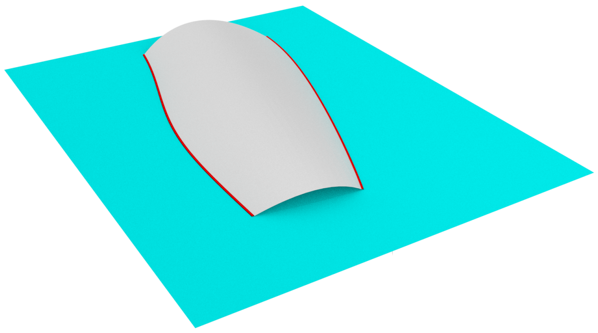
$$u_k = u_{k-1} + \delta u, \quad v_k = v_{k-1} + \delta v\,. \tag{3.19b}$$

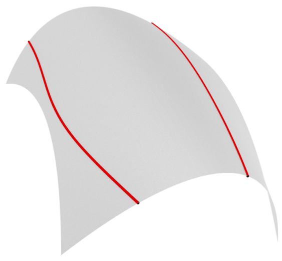
The marching steps associated with the starting point $\boldsymbol{P}_0$ stop when $\boldsymbol{P}_{\mathrm{A}}^{(k)}$ and $\boldsymbol{P}_{\mathrm{B}}^{(k)}$ are sufficiently close to each other, i.e. $\|\boldsymbol{P}_{\mathrm{A}}^{(k)} - \boldsymbol{P}_{\mathrm{B}}^{(k)}\| < \epsilon$, and the next point $\boldsymbol{P}_1$ found on the intersection curve is

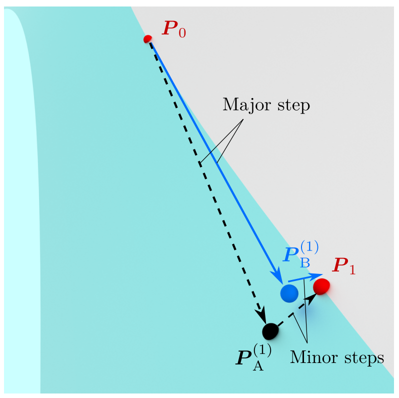
$$\boldsymbol{P}_1 = \frac{1}{2}(\boldsymbol{P}_{\mathrm{A}}^{(k)} + \boldsymbol{P}_{\mathrm{B}}^{(k)})\,. \tag{3.20}$$

The marching steps described above are repeated to find other points on the intersection curve until it reaches the finishing point on the intersection curve or a boundary edge.

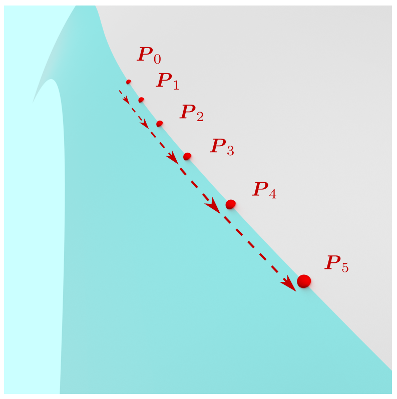
Figure 3.5 gives an example of the intersection between the plane and the same quadratic Bézier surface as in the divide-and-conquer method. Figure 3.5c illustrates the marching step to obtain the intersection point $\boldsymbol{P}_1$ from the starting point $\boldsymbol{P}_0$. $\boldsymbol{P}_{\mathrm{A}}^{(0)}$ and $\boldsymbol{P}_{\mathrm{B}}^{(0)}$ are points on the Bézier surface and the plane, respectively, after the major step, which are not coincident. A few subsequent minor steps are required to converge to the same point $\boldsymbol{P}_1$ which is approximately on the intersection curve within a tolerance. This marching process is repeated to obtain other points on the intersection curve as shown in Figure 3.5d. As a result, the marching method is more efficient in terms of computational time to obtain the intersection curve compared with the divide-and-conquer method. In addition, the intersection points between a starting point and a finishing point are automatically oriented. More examples of using the marching method to obtain intersection curves with different surfaces are shown in Figure 3.6, including the intersection of the same quadratic Bézier surface with a bi-linear Bézier surface, a bi-quadratic Bézier surface and a bi-cubic Bézier surface, respectively.


(a) The quadratic Bézier surface and the plane


(b) Intersection curve



(c) A major step and minor steps

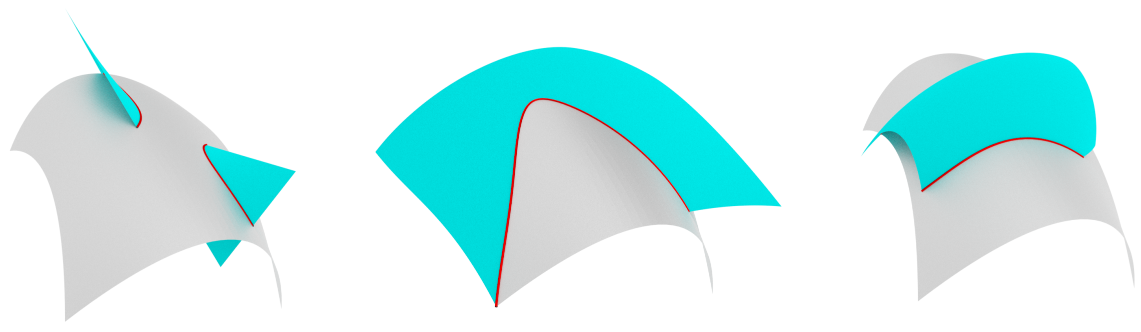


(d) Marching steps tracing the intersection curve

Fig. 3.5 Intersection of a quadratic Bézier surface with a plane using marching method.

3.1.3 Implicitisation

All parametric polynomial curves or surfaces have corresponding implicit representations. Using the implicitisation allows the reduction of the number of variables in the nonlinear equation which is usually required to be solved for the intersection problem. For simple geometries such as circles, spheres, etc., their implicit representations are straightforward to compute. However, it is in general difficult to compute the implicitisation of a free-form curve or surface. Algebraic geometry provides tools to obtain the implicit form of parametric curves or surfaces. The elimination methods which eliminate variables in the parametric form and obtain the implicit form based on the resultants of polynomials have been studied for decades [83–85]. Plane curves are well understood to obtain their implicitisation with the moving line method [48]. Nonetheless, it involves more effort to obtain the implicitisation of surfaces. In particular, only some special cases of surface parameterisation are possible to get a non-singular


(a) Bézier surfaces (blue) of degree 1×1, 2×2 and 3×3

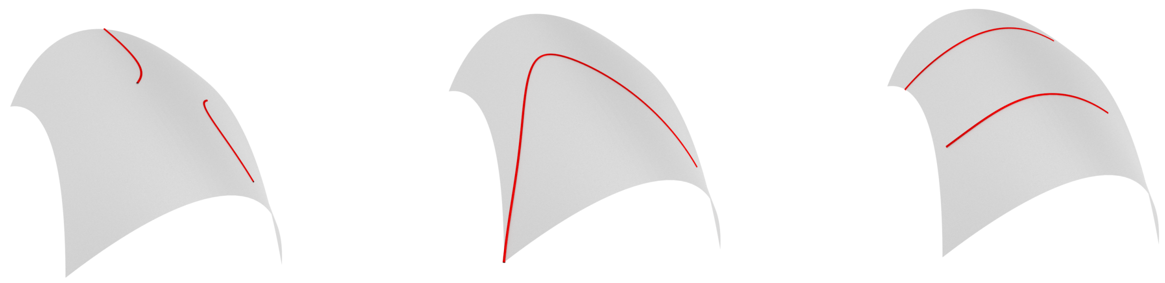

(b) Intersection curves

Fig. 3.6 Intersection of a bi-quadratic Bézier surface with a bi-linear Bézier surface, a bi-quadratic Bézier surface and a bi-cubic Bézier surface.

matrix with its determinant as the implicit equation of the surface. The approximate implicitisation method [86] was proposed to avoid the difficulty of implicitisation of higher-order parametric surfaces by approximating the surface with some lower-order implicit polynomial surfaces. Recently, a method of implicit matrix representations of parametric curves and surfaces has been proposed [50] to compute the exact implicitisation with matrix forms and the requirement of non-singular matrices is eliminated, which extends the implicitisation method to a broader class of parametric curves and surfaces. Both the elimination methods and the implicit matrix representations are exact implicitisation, which will be described in the following sections.

Elimination methods

The idea of elimination methods is to eliminate parametric variables in order to convert a parametric representation to an implicit form. For example, consider a parametric curve $\mathcal{C}(t)$ given by

$$\mathcal{C}(t) = (x(t), y(t)) = \left(t + 1, t^2 - 3t + 1\right) , \tag{3.21}$$

it is evident that t can be expressed in terms of x with $t = x - 1$, and substituting the expression of t into y yields

$$y = (x-1)^2 - 3(x-1) + 1 = x^2 - 5x + 5\,, \tag{3.22}$$

which is the implicit equation of the curve. In general, for curves with degree one or two it is straightforward to eliminate the variable in this way. However, for higher-degree curves and surfaces, it becomes difficult or even impossible using this direct elimination method. Hence, the concept of resultants used in algebraic geometry to find common roots is adopted to find the implicit form of higher-degree parameterisations.

Consider two polynomials $f(t)$ of degree m and $g(t)$ of degree n,

$$f(t) = f_0 + f_1 t + \cdots + f_m t^m \quad \text{and} \quad g(t) = g_0 + g_1 t + \cdots + g_n t^n\,, \tag{3.23}$$

the resultants of the two polynomials $f(t)$ and $g(t)$ indicate if they have common roots, that is, $f(t) = 0$ and $g(t) = 0$ intersect at some common values t_0. A set of auxiliary polynomials $h_j(t)$ in terms of $f(t)$ and $g(t)$ are constructed such that $h_j(t) = 0$ implies that $f(t) = 0$ and $g(t) = 0$. It is evident that $h_j(t) = 0$ leads to a homogeneous equation system, and the determinant of the coefficient matrix of this homogeneous equation system is computed as the resultant of the two polynomials $f(t)$ and $g(t)$, denoted with $R(f, g)$. Therefore, whether or not the two polynomials $f(t)$ and $g(t)$ have common roots can be determined by examining their resultant $R(f, g)$. The two polynomials have common roots only when $R(f, g) = 0$.

One way to construct the auxiliary polynomials $h_j(t)$ is to apply the Sylvester matrix of size $(m + n) \times (m + n)$ which is given by

$$\mathrm{Syl}(f,g) = \left[\begin{array}{cccccccc}
f_m & f_{m-1} & \cdots & f_0 & & & & \\
 & f_m & f_{m-1} & \cdots & f_0 & & & \\
 & & \ddots & \ddots & & \ddots & & \\
 & & & \ddots & \ddots & & \ddots & \\
 & & & & f_m & f_{m-1} & \cdots & f_0 \\
\hline
g_n & g_{n-1} & \cdots & \cdots & g_0 & & & \\
 & g_n & g_{n-1} & \cdots & \cdots & g_0 & & \\
 & & \ddots & \ddots & & & \ddots & \\
 & & & g_n & g_{n-1} & \cdots & \cdots & g_0
\end{array}\right]
\begin{array}{l}
\left.\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right\} \text{n} \\
\left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} \text{m}
\end{array}
. \tag{3.24}$$

The Sylvester matrix is constructed by elevating the powers of the polynomials with multiplying by t in turn until a square coefficient matrix is obtained. The resultant of the two polynomials is computed with $R(f,g) = \det(\mathrm{Syl}(f,g))$. Therefore, the two polynomials $f(t)$ and $g(t)$ have common roots if and only if the determinant of the Sylvester matrix (3.24) is zero. For example, the curve (3.21) can be denoted with the following two polynomials,

$$f(t) = t + 1 - x = 0 \quad \text{and} \quad g(t) = t^2 - 3t + 1 - y = 0\,, \tag{3.25}$$

and the Sylvester's resultant of these two polynomials is

$$R(f,g) = \det(\mathrm{Syl}(f,g)) = \begin{vmatrix} 1 & 1-x & 0 \\ 0 & 1 & 1-x \\ 1 & -3 & 1-y \end{vmatrix}\,. \tag{3.26}$$

The zero determinant gives the implicit equation of the curve, $y = x^2 - 5x + 5$, which is the same as (3.22).

Another approach to constructing the auxiliary polynomials $h_j(t)$ results in the Bézout matrix whose determinant can also be used as the resultant of the two polynomials. The auxiliary polynomials $h_j(t)$ are constructed as follows with $p = \max\{m, n\}$,

$$h_p(t) = g_p f(t) - f_p g(t)\,, \tag{3.27a}$$

$$h_{p-1}(t) = (g_p t + g_{p-1})\, f(t) - (f_p t + f_{p-1})\, g(t)\,, \tag{3.27b}$$

$$\vdots$$

$$\begin{aligned} h_1(t) = {} & (g_p t^{p-1} + g_{p-1} t^{p-2} + \cdots + g_1) f(t) \\ & - (h_p t^{p-1} + h_{p-1} t^{p-2} + \cdots + h_1) g(t)\,. \end{aligned} \tag{3.27c}$$

The coefficient matrix of the resulting homogeneous equation system $h_j(t) = 0$ yields the Bézout matrix $\mathrm{Bez}(f,g)$ of size $p \times p$ with entries b_{ij} given by

$$b_{ij} = \sum_{k=1}^{q} f_{j+k-1}\, g_{i-k} - f_{i-k}\, g_{j+k-1} \tag{3.28}$$

with $q = \min\{i, p+1-j\}$.

The resultant of the two polynomials $f(t)$ and $g(t)$ is computed with $R(f,g) = \det(\mathrm{Bez}(f,g))$. Hence, the two polynomials have common roots if and only if the determinant of the Bézout matrix is zero. As can be seen, even though the construction of the Sylvester matrix is more straightforward than the Bézout matrix, the size of the Bézout matrix is much smaller than the Sylvester matrix if the polynomial degree is high, which is advantageous since the determinant of the matrix needs to be computed. For the curve (3.21) the auxiliary polynomials $h_j(t)$ are

$$h_2(t) = t + 1 - x \quad \text{and} \quad h_1(t) = (1-x)t + 3x + y - 4\,, \tag{3.29}$$

which leads to the following Bézout resultant,

$$R(f,g) = \det(\mathrm{Bez}(f,g)) = \begin{vmatrix} 1 & 1-x \\ 1-x & 3x+y-4 \end{vmatrix}\,, \tag{3.30}$$

and its zero determinant leads to the same implicit equation $y = x^2 - 5x + 5$.

The resultant of two polynomials can be used to obtain the implicit form of a parameterisation. For example, consider a planar parametric curve $\mathcal{C}(t)$ with homogeneous coordinates $(X(t), Y(t), W(t))$ where $X(t)$, $Y(t)$ and $W(t)$ are polynomials, two polynomials can be constructed as

$$f(t) = xW(t) - X(t) \quad \text{and} \quad g(t) = yW(t) - Y(t)\,. \tag{3.31}$$

For any point t_0 on the curve, it must have $f(t_0) = 0$ and $g(t_0) = 0$. In other words, t_0 is a common root of the two polynomials $f(t)$ and $g(t)$. Therefore, the implicit equation of the curve $\mathcal{C}(t)$ is given by the zero resultant of the two polynomials $f(t)$ and $g(t)$, i.e. $\mathcal{C}(x,y) = R(f(t), g(t)) = 0$.

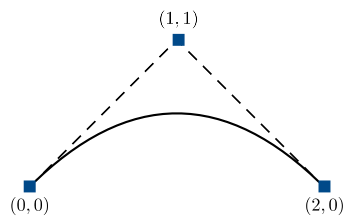
The moving line method proposed by Sederberg et al. [47, 48] gives a geometric interpretation for the implicitisation of planar curves, which employs the fact that any conic section can be generated by the intersection of two pencils of lines [49]. Figure 3.7 gives an example of tracing a planar quadratic Bézier curve with two pencils of lines. The basic idea of the moving line method is summarised as follows for planar curves.

Let $\boldsymbol{L}(t) = (a(t), b(t), c(t))$ denotes a pencil of lines

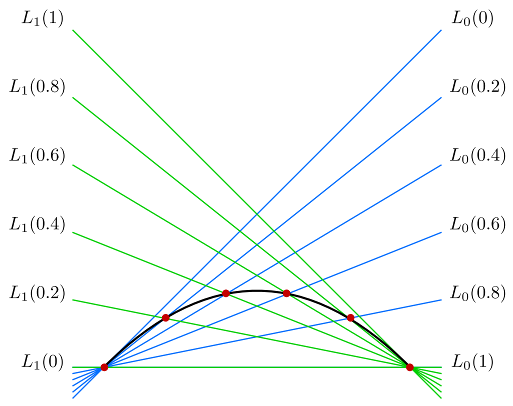
$$a(t)x + b(t)y + c(t) = 0\,, \tag{3.32}$$

where $a(t)$, $b(t)$ and $c(t)$ are linear functions of t. A moving point $\boldsymbol{P}(t)$ with the homogeneous coordinates $(X(t), Y(t), W(t))$ is said to follow the moving line $\boldsymbol{L}(t)$ if

$$\boldsymbol{P}(t) \cdot \boldsymbol{L}(t) = 0\,. \tag{3.33}$$



(a) A planar quadratic Bézier curve



(b) Two linear moving lines of the curve

Fig. 3.7 Linear moving lines of a planar quadratic curve.

Hence, two moving lines $\boldsymbol{L}_1(t)$ and $\boldsymbol{L}_2(t)$ intersect at a moving point $\boldsymbol{P}(t)$ if

$$\boldsymbol{P}(t) = \boldsymbol{L}_1(t) \times \boldsymbol{L}_2(t)\,, \tag{3.34}$$

and a moving line $\boldsymbol{L}(t)$ follows two moving points $\boldsymbol{P}_1(t)$ and $\boldsymbol{P}_2(t)$ if

$$\boldsymbol{L}(t) = \boldsymbol{P}_1(t) \times \boldsymbol{P}_2(t)\,, \tag{3.35}$$

where $\times$ denotes the cross product of two vectors.

As a result, if the two moving points are on the curve $\mathcal{C}(t)$, (3.35) leads to a matrix which is equivalent to Bézout matrix, as (3.33) gives a homogeneous equation system. In addition, it can be seen from the moving line method that the intersection points of two moving lines of degrees m and n respectively generate a curve of degree $m+n$ in general when base points do not occur. For example, a quadratic curve can be generated by the intersection of two linear pencils of lines, as illustrated in Figure 3.7.

The implicitisation using resultants of polynomials can be extended to obtain the implicit equations for space curves and surfaces. For a curve the degree of its implicit equation is the same as its parameterisation. For a triangular surface patch of degree n, the degree of its implicit equation is in general n^2; for a tensor-product surface patch of bi-degree (m, n), the degree of its implicit equation is generally $2mn$, indicating that the size of the Sylvester matrix or the Bézout matrix can be very large for surfaces. Furthermore, the implicit expression requires the computation of the determinant of the matrix, which is not cheap to compute for a matrix with a large size. Hence, the high computation cost for high-degree parameterisations is one of the problems when using elimination methods.

Another difficulty with elimination methods is the occurrence of base points where all the components of the parameterisation are zero, for example, $X(t) = 0$, $Y(t) = 0$ and $W(t) = 0$ for a planar curve, when the traditional resultant approach does not apply. However, base points are quite common, especially in free-form surfaces. As a matter of fact, it is only possible for some particular classes of parameterisation to get non-singular matrices with the zero determinant as the implicit equation [87–89].

Implicit matrix representations

The elimination methods aim to obtain the implicit equations from computing the determinant of non-singular matrices. However, a non-singular matrix cannot be always derived for free-form surfaces. In order to eliminate the requirement of non-singular matrices, one can keep the implicitisation in matrix form, which results in the implicit matrix representations (M-Reps) of parameterisations. In this method the matrix is not required to be non-singular, and the rank of the implicit M-Rep drops if the point evaluated is exactly on the curve or surface, which can be determined from the singular value decomposition (SVD) of the implicit M-Rep, avoiding the determinant computation of a matrix.

As described in Section 2.3, subdivision surfaces can be converted to a collection of Bézier surfaces. It is general to consider the implicit matrix representations of Bézier curves and surfaces, and the subdivision surface becomes a collection of these implicit representations of Bézier surfaces.

Consider a rational Bézier representation in homogeneous coordinates

$$\boldsymbol{f}(\boldsymbol{\theta}) = \begin{pmatrix} w\boldsymbol{x} \\ w \end{pmatrix} = \sum_{\boldsymbol{i}} B_{\boldsymbol{i}}^{\boldsymbol{\mu}}(\boldsymbol{\theta}) \begin{pmatrix} w_{\boldsymbol{i}}\boldsymbol{x}_{\boldsymbol{i}} \\ w_{\boldsymbol{i}} \end{pmatrix} , \tag{3.36}$$

where $\boldsymbol{x}_{\boldsymbol{i}}$ are the control coordinates in $\mathbb{R}^3$ and $w_{\boldsymbol{i}}$ the associated weights. The indices $\boldsymbol{i}$ and $\boldsymbol{\mu}$ are again multi-indices for surfaces. The Bernstein polynomials $B_{\boldsymbol{i}}^{\boldsymbol{\mu}}(\boldsymbol{\theta})$ have the same expressions as in Section 2.3 for curves and surfaces.

We consider an auxiliary vector of polynomials

$$\boldsymbol{g}(\boldsymbol{\theta}) = \sum_{\boldsymbol{j}} \widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}) \gamma_{\boldsymbol{j}} \tag{3.37}$$

with Bézier basis functions $\widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}$, coefficients $\boldsymbol{\gamma_j}$ which are yet to be determined, and the index $\boldsymbol{j}$ a multi-index for surfaces. Each of the coefficients $\boldsymbol{\gamma_j} = \left(\gamma_{\boldsymbol{j}}^1, \gamma_{\boldsymbol{j}}^2, \gamma_{\boldsymbol{j}}^3, \gamma_{\boldsymbol{j}}^4\right)^{\mathsf{T}}$ is in $\mathbb{R}^4$. The degree $\boldsymbol{\nu}$ of the Bézier basis $\widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}$ is chosen according to [51], and the number of basis functions n_j is determined accordingly. For a Bézier curve of degree μ, $\nu = \mu - 1$, $n_j = \nu + 1 = \mu$; for a tensor-product Bézier surface of bi-degree, $\boldsymbol{\nu} = (\nu_1, \nu_2) = (2\mu_1 - 1, \mu_2 - 1)$ or $(\mu_1 - 1, 2\mu_2 - 1)$; for a triangular Bézier surface of degree μ, $\nu = 2(\mu - 1)$. Therefore, for a bi-cubic Bézier surface (corresponding to the regular Catmull-Clark subdivision patch), $\boldsymbol{\nu} = (\nu_1, \nu_2) = (5, 2)$ or $(2, 5)$, $n_{\boldsymbol{j}} = (\nu_1 + 1)(\nu_2 + 1) = 18$; for a quartic triangular Bézier surface (corresponding to the regular Loop subdivision patch), $\nu = 6$, $n_{\boldsymbol{j}} = (\nu + 2)(\nu + 1)/2 = 28$. In the following context, we consider a single Bézier patch $\boldsymbol{f}(\boldsymbol{\theta})$ of bi-degree $\boldsymbol{\mu} = (p, p)$ as in applications the same degree is usually used for tensor-product spline surfaces. It is worth noting that the degree $\boldsymbol{\nu}$ can be chosen to be higher than the values suggested above, which will result in a larger matrix as will be introduced later and hence is not advantageneous for the intersection computation.

The two vectors $\boldsymbol{f}(\boldsymbol{\theta})$ and $\boldsymbol{g}(\boldsymbol{\theta})$ each with four components are required to be orthogonal

$$\boldsymbol{f}(\boldsymbol{\theta}) \cdot \boldsymbol{g}(\boldsymbol{\theta}) = 0\,, \tag{3.38}$$

that is,

$$\left(\sum_{\boldsymbol{i}} B_{\boldsymbol{i}}^{\boldsymbol{\mu}}(\boldsymbol{\theta}) \begin{pmatrix} w_{\boldsymbol{i}} \boldsymbol{x}_{\boldsymbol{i}} \\ w_{\boldsymbol{i}} \end{pmatrix} \right) \cdot \left(\sum_{\boldsymbol{j}} \widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}) \boldsymbol{\gamma_j} \right) = 0\,. \tag{3.39}$$

Evidently, the product of two Bézier basis functions can be again expressed with a Bézier basis function. Hence, (3.39) can be written with a new Bézier basis $\widehat{B}_{\boldsymbol{k}}$ of bi-degree $(3p - 1, 2p - 1)$ as

$$\sum_{\boldsymbol{j}} \sum_{\boldsymbol{k}} \widehat{B}_{\boldsymbol{k}} C_{\boldsymbol{kj}} \boldsymbol{\gamma_j} = 0\,, \tag{3.40}$$

where $C_{\boldsymbol{kj}}$ are the coefficients of a matrix $\boldsymbol{C} \in \mathbb{R}^{6p^2 \times 8p^2}$. Note that there are $6p^2$ basis functions in the basis with bi-degree $(3p - 1, 2p - 1)$, $2p^2$ basis functions in the basis with bi-degree $(2p - 1, p - 1)$ and each $\boldsymbol{\gamma_j}$ has four coefficients. The matrix $\boldsymbol{C}$ contains the control point coordinates, associated weights and the basis transformation coefficients resulting from the Bézier basis multiplication.

The orthogonality constraint (3.38) implies $6p^2$ equations for determining $8p^2$ unknown components of $\boldsymbol{\gamma_j}$'s so that the matrix $\boldsymbol{C}$ must be rank deficient. The non-trivial $\boldsymbol{\gamma_j}$'s satisfying (3.40) must lie in the null space of $\boldsymbol{C}$, which is straightforward to compute with a singular value decomposition (SVD). The singular value decomposition of $\boldsymbol{C}$ reads

$$\boldsymbol{C} = \boldsymbol{U}_c \boldsymbol{\Sigma}_c \boldsymbol{V}_c^{\mathsf{T}} \tag{3.41}$$

where $\boldsymbol{U}_c$ is a matrix of left singular vectors, $\boldsymbol{\Sigma}_c$ is a diagonal matrix of singular values and $\boldsymbol{V}_c$ is a matrix of right singular vectors. The singular values in $\boldsymbol{\Sigma}_c$ are usually sorted in descending order starting from the top diagonal. The right null vectors are the columns of $\boldsymbol{V}_c$ which correspond to zero diagonal entries in $\boldsymbol{\Sigma}_c$. For a specific Bézier surface $\boldsymbol{f}(\boldsymbol{\theta})$, the number of right null vectors n_i of $\boldsymbol{C}$, $\dim(\mathrm{Null}(\boldsymbol{C}))$,depends on the coordinates of its control points and the associate weights.

Denoting the right null vectors of $\boldsymbol{C}$ with $\boldsymbol{\gamma}_{\boldsymbol{j}}^{(i)}$ and introducing them into the auxiliary vector of polynomials (3.37) yields

$$\boldsymbol{g}^{(i)}(\boldsymbol{\theta}) = \sum_{\boldsymbol{j}} \widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}) \boldsymbol{\gamma}_{\boldsymbol{j}}^{(i)} \tag{3.42}$$

For a fixed parametric coordinate $\boldsymbol{\theta}^*$, a set of planes are defined by the vectors $\boldsymbol{g}^{(i)}(\boldsymbol{\theta}^*)$ with the implicit equations

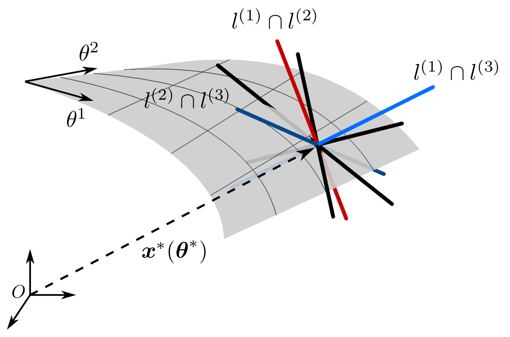
$$l^{(i)}(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{g}^{(i)}(\boldsymbol{\theta}^*) \cdot \begin{pmatrix} w\boldsymbol{x} \\ w \end{pmatrix} = 0 \,, \quad \text{that is,} \quad l^{(i)}(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{g}^{(i)}(\boldsymbol{\theta}^*) \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} = 0 \,. \tag{3.43}$$

Due to the orthogonality condition (3.38) the surface point with the homogeneous coordinate $\boldsymbol{f}(\boldsymbol{\theta}^*)$ and the surface coordinate

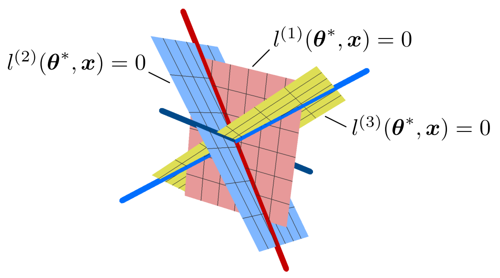
$$\boldsymbol{x}^* = \left(\frac{f_1(\boldsymbol{\theta}^*)}{f_4(\boldsymbol{\theta}^*)}, \frac{f_2(\boldsymbol{\theta}^*)}{f_4(\boldsymbol{\theta}^*)}, \frac{f_3(\boldsymbol{\theta}^*)}{f_4(\boldsymbol{\theta}^*)} \right)^{\mathsf{T}}$$

satisfies all plane equations and, hence, is present on all planes, i.e. $\boldsymbol{x}^* \in l^{(i)}(\boldsymbol{\theta}^*, \boldsymbol{x}^*) = 0 \;\forall\; i$. For a point $\boldsymbol{x}^*$ to be present on two or more planes it must lie on their intersection. Note that the intersection of two planes defines a line and the intersection of a plane with a line defines a point. These observations motivate the definition of the surface $\boldsymbol{f}(\boldsymbol{\theta})$ as the intersection of a set of planes. Evidently, the planes must move, or change their inclination, while the parametric coordinate $\boldsymbol{\theta}$ is varied. Hence, each $l^{(i)}(\boldsymbol{\theta}, \boldsymbol{x}) = 0$ for a fixed i represents a family of planes and is traditionally referred to as a *moving plane* or a *pencil of planes* [47].

Although three planes are sufficient to define a point, according to [51] more than three planes must be used, in general, to describe an entire patch. As an example, Figure 3.8 gives an example of a Bézier surface traced by four planes. The four planes



(a) Intersection lines of four planes $l^{(i)}(\boldsymbol{\theta}^*, \boldsymbol{x})$


$l^{(1)}(\boldsymbol{\theta}^*, \boldsymbol{x}) = 0$
$l^{(2)}(\boldsymbol{\theta}^*, \boldsymbol{x}) = 0$
$l^{(3)}(\boldsymbol{\theta}^*, \boldsymbol{x}) = 0$

(b) Three of the four planes with their intersection lines

Fig. 3.8 Intersection of planes $l^{(i)}(\boldsymbol{\theta}^*, \boldsymbol{x})$ tracing the surface.

intersect at a point which is exactly on the surface (Figure 3.8a). For the sake of clarity, in Figure 3.8b only three of the planes are depicted. Similarly, the planes are degenerated to lines for curves, and in the space at least two lines are required to define a point on a curve. As illustrated in Figure 3.9, for a cubic Bézier curve evaluated at different parameters θ^*, the three lines move accordingly but always intersect exactly on the curve as shown in Figure 3.9b; in other words, the curve is traced by the intersection points of the three moving lines.

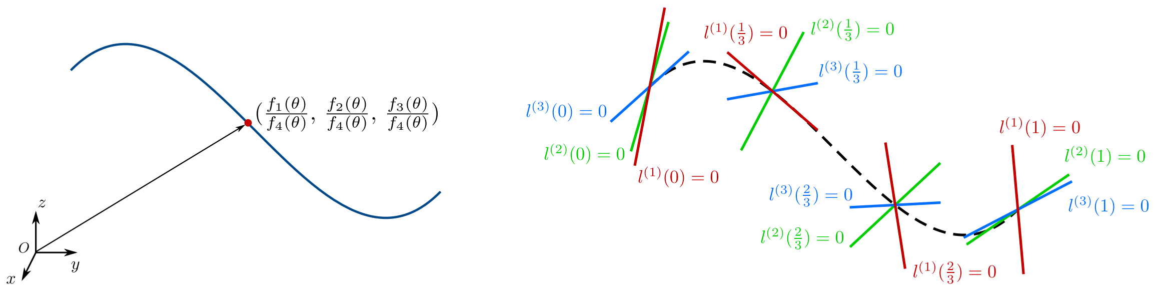
Substituting (3.42) into the plane equation (3.43) yields

$$l^{(i)}(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{g}^{(i)}(\boldsymbol{\theta}^*) \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} = \sum_{\boldsymbol{j}} \widetilde{B}^{\boldsymbol{\nu}}_{\boldsymbol{j}}(\boldsymbol{\theta}^*) \left(\boldsymbol{\gamma}^{(i)}_{\boldsymbol{j}} \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \right) = \sum_{\boldsymbol{j}} \widetilde{B}^{\boldsymbol{\nu}}_{\boldsymbol{j}}(\boldsymbol{\theta}^*) M^{(i)}_{\boldsymbol{j}}(\boldsymbol{x}) = 0 \,, \tag{3.44}$$

where $M^{(i)}_{\boldsymbol{j}}(\boldsymbol{x})$ constitutes the i-th column of a matrix $\boldsymbol{M}(\boldsymbol{x})$, that is,

$$\boldsymbol{M}(x, y, z) = \begin{pmatrix} M^{(1)}_{\boldsymbol{1}} & M^{(2)}_{\boldsymbol{1}} & \cdots & M^{(n_i)}_{\boldsymbol{1}} \\ M^{(1)}_{\boldsymbol{2}} & M^{(2)}_{\boldsymbol{2}} & \cdots & M^{(n_i)}_{\boldsymbol{2}} \\ \vdots & \vdots & & \vdots \\ M^{(1)}_{\boldsymbol{n_j}} & M^{(2)}_{\boldsymbol{n_j}} & \cdots & M^{(n_i)}_{\boldsymbol{n_j}} \end{pmatrix} \tag{3.45}$$

with the index $\boldsymbol{j}$ of $M^{(i)}_{\boldsymbol{j}}$ a multi-index for surfaces. The number of columns n_i of $\boldsymbol{M}(\boldsymbol{x})$ is equal to the number of planes or null vectors $\boldsymbol{\gamma}^{(i)}_{\boldsymbol{j}}$, and its number of rows is $n_{\boldsymbol{j}} = 2p^2$. Again, due to the orthogonality condition (3.38), for a point with the homogeneous coordinate $\boldsymbol{f}(\boldsymbol{\theta}^*)$ and the surface coordinate $\boldsymbol{x}^*$, the set of equations (3.44) are satisfied for all i. This implies that there has to be a change in the rank of $\boldsymbol{M}(\boldsymbol{x})$ when evaluated


$(\frac{f_1(\theta)}{f_4(\theta)}, \frac{f_2(\theta)}{f_4(\theta)}, \frac{f_3(\theta)}{f_4(\theta)})$
z
O
y
x
$l^{(1)}(\frac{1}{3}) = 0$
$l^{(2)}(\frac{1}{3}) = 0$
$l^{(3)}(\frac{1}{3}) = 0$
$l^{(3)}(0) = 0$
$l^{(2)}(0) = 0$
$l^{(1)}(0) = 0$
$l^{(1)}(1) = 0$
$l^{(2)}(1) = 0$
$l^{(3)}(1) = 0$
$l^{(3)}(\frac{2}{3}) = 0$
$l^{(2)}(\frac{2}{3}) = 0$
$l^{(1)}(\frac{2}{3}) = 0$

(a) A cubic curve in the space (b) The intersection of $l^{(i)}(\theta, x)$ tracing the curve

Fig. 3.9 $l^{(i)}(\boldsymbol{\theta}, \boldsymbol{x})$ of a cubic curve in the Cartesian coordinate system.

at $\boldsymbol{x}^*$. Furthermore, the minimum degree $\boldsymbol{\nu} = (2p-1, p-1)$ suggested for the basis $\widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}$ ensures that $\boldsymbol{M}(\boldsymbol{x})$ is full rank except on the surface [51]. This motivates the use of the change of rank of $\boldsymbol{M}(\boldsymbol{x})$ to define the surface $\boldsymbol{f}(\boldsymbol{\theta})$, which gives rise to the notion of matrix representation (M-Rep) of a spline surface.

Alternatively, $\boldsymbol{\gamma}_{\boldsymbol{j}}^{(i)}$ can be further rearranged such that it is of the form

$$\begin{pmatrix}\gamma_{\mathbf{1}}^{1(i)} & \cdots & \gamma_{\boldsymbol{n_j}}^{1(i)} & \gamma_{\mathbf{1}}^{2(i)} & \cdots & \gamma_{\boldsymbol{n_j}}^{2(i)} & \gamma_{\mathbf{1}}^{3(i)} & \cdots & \gamma_{\boldsymbol{n_j}}^{3(i)} & \gamma_{\mathbf{1}}^{4(i)} & \cdots & \gamma_{\boldsymbol{n_j}}^{4(i)}\end{pmatrix}^{\mathsf{T}} . \tag{3.46}$$

The null space of the matrix $\boldsymbol{C}$ is then denoted as

$$\mathrm{Null}(\boldsymbol{C}) = \begin{pmatrix}\boldsymbol{\gamma}_{\boldsymbol{j}}^{(1)} & \boldsymbol{\gamma}_{\boldsymbol{j}}^{(2)} & \cdots & \boldsymbol{\gamma}_{\boldsymbol{j}}^{(n_i)}\end{pmatrix} = \begin{pmatrix}\boldsymbol{M}^1 \\ \boldsymbol{M}^2 \\ \boldsymbol{M}^3 \\ \boldsymbol{M}^4\end{pmatrix} . \tag{3.47}$$

Hence, the implicit matrix representation $\boldsymbol{M}$ can also be obtained with

$$\boldsymbol{M} = x\boldsymbol{M}^1 + y\boldsymbol{M}^2 + z\boldsymbol{M}^3 + \boldsymbol{M}^4 . \tag{3.48}$$

Properties of implicit matrix representations

With the implicit matrix representation (3.45), the equation (3.44) can be expressed in matrix form as

$$\begin{aligned}\begin{pmatrix}l^{(1)} & l^{(2)} & \cdots & l^{(n_i)}\end{pmatrix} &= \begin{pmatrix}\widetilde{B}_{\mathbf{1}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}^*) & \widetilde{B}_{\mathbf{2}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}^*) & \cdots & \widetilde{B}_{\boldsymbol{n_j}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}^*)\end{pmatrix} \boldsymbol{M}(x, y, z) \\ &= \begin{pmatrix}0 & 0 & \cdots & 0\end{pmatrix} ,\end{aligned} \tag{3.49}$$

and this expression holds only when it is evaluated at the point on the parameterisation $\boldsymbol{f}(\boldsymbol{\theta})$. It can be observed from (3.49) that the matrix $\boldsymbol{M}(x, y, z)$ must be rank deficient in order to have non-trivial solutions, since the basis $\{\widetilde{B}^{\boldsymbol{\nu}}_{\boldsymbol{j}}(\boldsymbol{\theta}^*)\}(\boldsymbol{j} = \boldsymbol{1}, \cdots, \boldsymbol{n_j})$ cannot be a zero vector. In other words, the rank of the matrix $\boldsymbol{M}$ drops only at points on the curve or surface studied. If the point is not on the curve or surface, the matrix $\boldsymbol{M}$ has a full rank.

The drop-of-rank property of the matrix $\boldsymbol{M}$ indicates that it can be used as an implicit representation of the parameterisation $\boldsymbol{f}(\boldsymbol{\theta})$ [51]. The implicitisation property of the matrix $\boldsymbol{M}$ can be further examined by defining an evaluation function

$$\delta_{\boldsymbol{M}}(x, y, z) = \prod_{i=1}^{n_j} \sigma_i \left(\boldsymbol{M}(x, y, z)\right) , \tag{3.50}$$

where σ_i are singular values of the matrix $\boldsymbol{M}$ evaluated at a point (x, y, z). If the matrix $\boldsymbol{M}$ is not with a full rank, $\delta_{\boldsymbol{M}}(x, y, z) = 0$; otherwise, it has a positive value. In other words, a point (x^*, y^*, z^*) belongs to $\boldsymbol{f}(\boldsymbol{\theta})$ only if $\delta_{\boldsymbol{M}}(x^*, y^*, z^*) = 0$, which resembles a distance function of the curve or surface evaluated.

In addition, the parameter(s) $\boldsymbol{\theta}^*$ corresponding to the point (x^*, y^*, z^*) on the parameterisation $\boldsymbol{f}(\boldsymbol{\theta})$ can be inferred from the equation (3.49). Since the basis $\{\widetilde{B}^{\boldsymbol{\nu}}_{\boldsymbol{j}}(\boldsymbol{\theta}^*)\}$ forms the left null space of the matrix $\boldsymbol{M}$, the parameter(s) $\boldsymbol{\theta}^*$ can be obtained by computing the ratio of pairs of the basis functions, which yields linear equations that can be trivially solved. This inversion property of the implicit matrix representation is useful in the intersection computation to compute the surface parametric coordinates of intersection points.

Examples

A cubic Bézier curve Consider a cubic Bézier curve with control points $\boldsymbol{x}_1 = (0, 0, 0)$, $\boldsymbol{x}_2 = (1, 2, 0)$, $\boldsymbol{x}_3 = (2, -1, 0)$ and $\boldsymbol{x}_4 = (4, 0, 0)$, and the weights of the control points are all equal to 1. The plot of the cubic Bézier curve is shown in Figure 3.10a.

The degree of the auxiliary polynomials $\boldsymbol{g}(\boldsymbol{\theta})$ is chosen as 2 and expressed with quadratic Bézier basis functions, that is,

$$\boldsymbol{g}(\boldsymbol{\theta}) = (1 - \theta)^2 \boldsymbol{\gamma}_1 + 2\theta(1 - \theta)\boldsymbol{\gamma}_2 + \theta^2 \boldsymbol{\gamma}_3 \tag{3.51}$$

with each $\boldsymbol{\gamma}_i$ in $\mathbb{R}^4$.

After substituting the auxiliary vector $\boldsymbol{g}(\boldsymbol{\theta})$ into the orthogonality condition (3.38), the coefficient matrix $\boldsymbol{C} \in \mathbb{R}^{6\times 12}$ is then given by, according to (3.40),

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0.6 & 0 & 0 & 1.2 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0.4 & 0 \\ 0.6 & 0.6 & 0 & -0.3 & 1.2 & 0 & 0 & 0 & 0 & 0.3 & 0.6 & 0.1 \\ 0.4 & 1.2 & 0.3 & 0 & -0.6 & 0.6 & 0 & 0 & 0 & 0.1 & 0.6 & 0.3 \\ 0 & 1.6 & 1.2 & 0 & 0 & -0.6 & 0 & 0 & 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.52}$$

The null space of the matrix $\boldsymbol{C}$ gives six independent vectors of the unknown coefficients of the auxiliary polynomials as follows,

$$\begin{aligned} \boldsymbol{\gamma} &= \begin{pmatrix} \boldsymbol{\gamma}^{(1)} & \boldsymbol{\gamma}^{(2)} & \boldsymbol{\gamma}^{(3)} & \boldsymbol{\gamma}^{(4)} & \boldsymbol{\gamma}^{(5)} & \boldsymbol{\gamma}^{(6)} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 0 & -0.745895 & -0.231156 & -0.0326366 \\ 0 & 0 & 0 & 0.1405 & -0.313385 & -0.146533 \\ 0 & 0 & 0 & 0.0112503 & -0.00442619 & -0.238746 \\ 0 & 0 & 0 & 0.331369 & -0.166017 & 0.033496 \\ 0 & 0 & 0 & 0.326922 & -0.193102 & 0.0441434 \\ 0 & 0 & 0 & 0.435324 & -0.263651 & 0.0523816 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.124736 & 0.844785 & -0.0515331 \\ 0 & 0 & 0 & -0.0450013 & 0.0177047 & 0.954983 \end{pmatrix} \end{aligned} \tag{3.53}$$

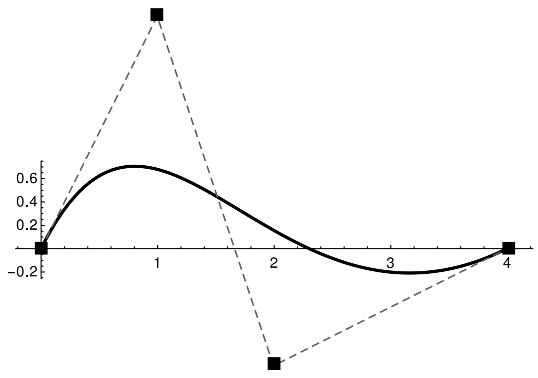
with $\boldsymbol{\gamma}_{\boldsymbol{j}}^{(i)}$ arranged in the form as (3.46), that is,

$$\boldsymbol{\gamma}_{\boldsymbol{j}}^{(i)} = \begin{pmatrix} \gamma_{1,1}^{(i)} & \gamma_{2,1}^{(i)} & \gamma_{3,1}^{(i)} & \gamma_{1,2}^{(i)} & \gamma_{2,2}^{(i)} & \gamma_{3,2}^{(i)} & \gamma_{1,3}^{(i)} & \gamma_{2,3}^{(i)} & \gamma_{3,3}^{(i)} & \gamma_{1,4}^{(i)} & \gamma_{2,4}^{(i)} & \gamma_{3,4}^{(i)} \end{pmatrix}^{\mathsf{T}}. \tag{3.54}$$

Hence, the resulting six auxiliary vectors of polynomials can be identified with six lines. Three of the lines are plotted in Figure 3.10b since the other three lines correspond to the trivial $z = 0$.

Therefore, the implicit matrix representation of the cubic Bézier curve is obtained as follows with six significant digits,

$$\boldsymbol{M}^{\mathsf{T}}(x,y,z) =$$
$$\begin{bmatrix} z & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & z \\ -0.745895x + 0.331369y & 0.124736 + 0.1405x + 0.326922y & -0.0450013 + 0.0112503x + 0.435324y \\ -0.231156x - 0.166017y & 0.844785 - 0.313385x - 0.193102y & 0.0177047 - 0.00442619x - 0.263651y \\ -0.0326366x + 0.033496y & -0.0515331 - 0.146533x + 0.0441434y & 0.954983 - 0.238746x + 0.0523816y \end{bmatrix}. \tag{3.55}$$



(a) The cubic Bézier curve

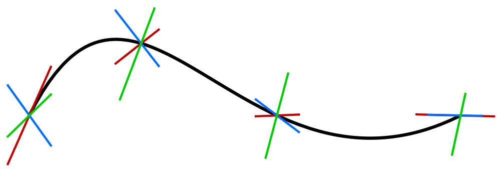

(b) The three moving lines of the curve

Fig. 3.10 The cubic Bézier curve and its corresponding auxiliary polynomials.

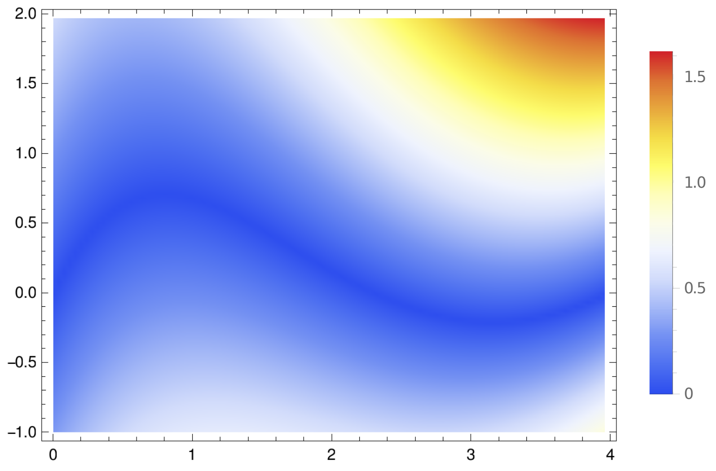
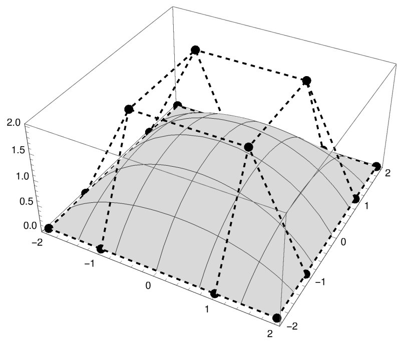


Fig. 3.11 Distance contour of the cubic Bézier curve.

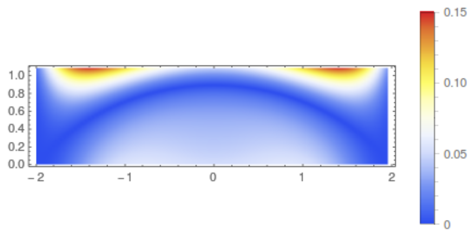
Figure 3.11 gives the distance contour of the cubic Bézier curve evaluated with the distance function (3.50). It can be seen from the contour plot that the zero contour reveals the cubic Bézier curve. Any point that is not on the curve has a non-zero distance value, and all the points with zero distance value form the curve.

A bi-cubic Bézier surface Consider a bi-cubic Bézier surface defined with control points $\boldsymbol{x}_{11} = (2,-2,0)$, $\boldsymbol{x}_{12} = (2,-1,0)$, $\boldsymbol{x}_{13} = (2,1,0)$, $\boldsymbol{x}_{14} = (2,2,0)$, $\boldsymbol{x}_{21} = (1,-2,0)$, $\boldsymbol{x}_{22} = (1,-1,2)$, $\boldsymbol{x}_{23} = (1,1,2)$, $\boldsymbol{x}_{24} = (1,2,0)$, $\boldsymbol{x}_{31} = (-1,-2,0)$, $\boldsymbol{x}_{32} = (-1,-1,2)$, $\boldsymbol{x}_{33} = (-1,1,2)$, $\boldsymbol{x}_{34} = (-1,2,0)$, $\boldsymbol{x}_{41} = (-2,-2,0)$, $\boldsymbol{x}_{42} = (-2,-1,0)$, $\boldsymbol{x}_{43} = (-2,1,0)$ and $\boldsymbol{x}_{44} = (-2,2,0)$. The weights of control points are all equal to 1.

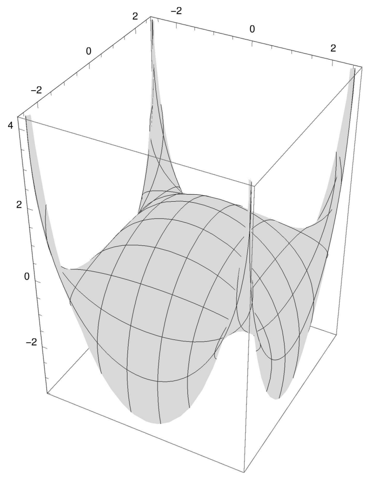
Figure 3.12a is the plot of the bi-cubic Bézier surface in the parametric domain $[0,1] \times [0,1]$. As the implicit equation of the Bézier surface is for the whole space, the Bézier surface in the parametric domain $[-0.5,1.5] \times [-0.5,1.5]$ is also given in Figure 3.12c as a reference.



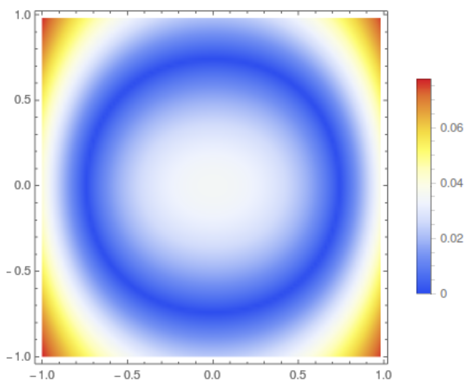
(a) The Bézier surface in the parametric domain $[0,1] \times [0,1]$



(b) The distance contour on plane $y = 1$



(c) The surface in the parametric domain $[-0.5,1.5] \times [-0.5,1.5]$



(d) The distance contour on plane $z = 1$

Fig. 3.12 The bi-cubic Bézier surface and its distance contours.

In order to construct the implicit matrix representation of the bi-cubic Bézier surface, the auxiliary polynomials are chosen in terms of Bézier basis functions of degrees 5×2. Expanding equation (3.39) gives

$$\sum_{k=1}^{4}\sum_{j_1=1}^{6}\sum_{j_2=1}^{3}\gamma_{j_1,j_2}^{k}\sum_{i_1=1}^{4}\sum_{i_2=1}^{4}P_{i_1,i_2}^{k}\frac{\binom{3}{i_1-1}\binom{3}{i_2-1}\binom{5}{j_1-1}\binom{2}{j_2-1}}{\binom{8}{i_1+j_1-2}\binom{5}{i_2+j_2-2}}B_{i_1+j_1-1}^{8}(u)B_{i_2+j_2-1}^{5}(v)=0\,, \tag{3.56}$$

where $\boldsymbol{P}_{i_1,i_2}$ are the homogeneous coordinates of control points and k denotes the k-th component of the coordinate. The above equation can be rewritten in matrix form as

$$\begin{bmatrix}B_1^8(u)B_1^5(v) & \cdots & B_9^8(u)B_6^5(v)\end{bmatrix}\boldsymbol{C}\begin{bmatrix}\gamma_{1,1}^1 & \cdots & \gamma_{6,3}^4\end{bmatrix}^{\mathsf{T}}=0 \tag{3.57}$$

with the coefficient matrix $\boldsymbol{C}$ of size 54×72.

The resulting implicit matrix $\boldsymbol{M}$ is of size 18×23. Figure 3.12b and 3.12d show the distance contours of the bi-cubic Bézier surface on the plane $y = 1$ and $z = 1$, respectively, which reflect the surface cut by the two planes.

3.2 Line/subdivision surface intersection

3.2.1 Hierarchical bounding volume trees

As mentioned a subdivision surface can be seen as a collection of patches, each of which is equivalent to a parametric surface piece. In other words, the subdivision surface is a piecewise parametric surface. For each patch of the subdivision surface, it has a convex hull property, that is, the limit surface represented by the patch is contained entirely within a convex hull which is defined by the patch. Therefore, the k-dops of the subdivision surface can be used to check the possibility of intersections as introduced in Section 3.1.1 for the divide-and-conquer method.

For the line/subdivision surface intersection, it is possible to check the intersection of a line with each patch in the subdivision surface. However, this approach is time-consuming and becomes inefficient for a large number of lines and patches. Therefore, a hierarchical bounding volume tree data structure is constructed to identify the potential intersection between the line and the patches.

First of all, the tree is created in an octree manner by recursively splitting the facets of the subdivision surface into eight subsets starting from the entire surface. In each step the splitting is based on the coordinates of the patch centroids which are computed as the average of vertex coordinates. A schematic octree is sketched in Figure 3.13. The root of the tree contains all the facets of the subdivision surface. Each node in the tree is a collection of facets containing their bounding volume information.

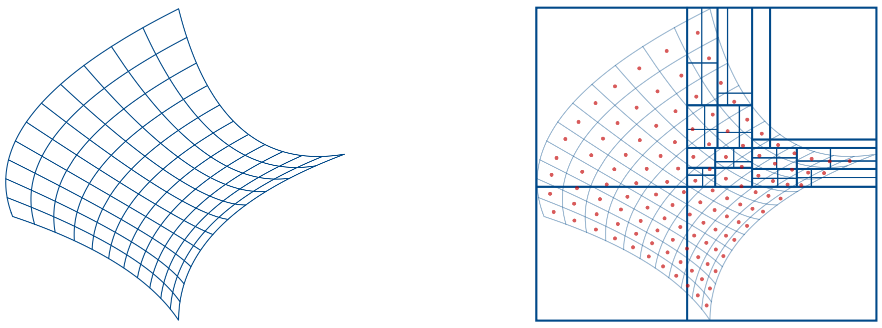

(a) The control mesh (b) An octree of the surface

Fig. 3.13 A schematic space division of a subdivision surface.

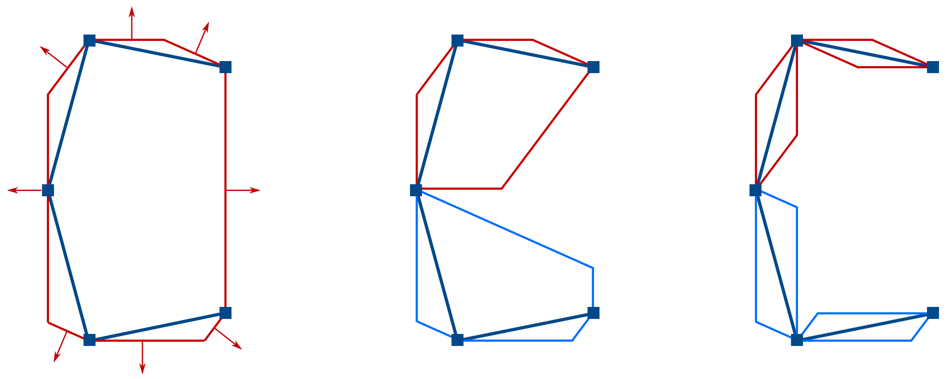

(a) Root (b) Level 1 (c) Level 2

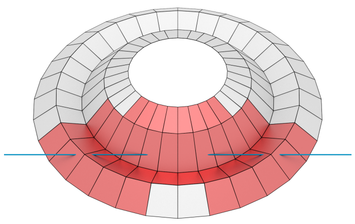
Fig. 3.14 A schematic bounding volume tree of k-dops.

The bounding volume of each node is represented with the k-dop of the set of patches belonging to the node. By constructing in this way, each leaf at the bottom of the octree contains only one facet in the subdivision surface, and the k-dops of these leaf nodes can be computed readily with (3.1).

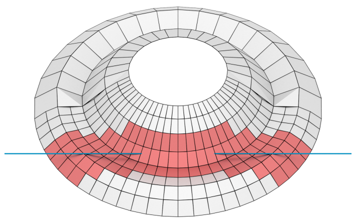
For the computation of k-dops of nodes in the tree, it is again not necessary to explicitly construct them. It is sufficient to consider all the vertices belonging to the node, which is similar to the single patch case discussed in preceding Section 3.1.1. Alternatively, the support heights for each node can be efficiently computed by starting from the tree leaves and computing the support heights of parent nodes from child nodes,

$$h_{j,\max}^{\text{parent}} = \max_{\text{child}} h_{j,\max}^{\text{child}} \quad \text{and} \quad h_{j,\min}^{\text{parent}} = \min_{\text{child}} h_{j,\min}^{\text{child}} \,. \tag{3.58}$$

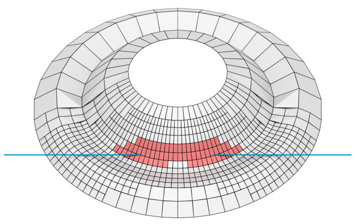
In the implemented k-dop bounding volume tree, each node represents the k-dop of either one single or a set of patches, see Figure 3.14.



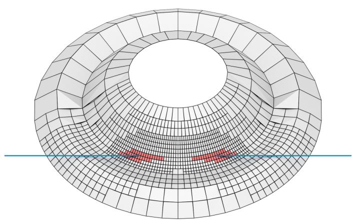
(a) Facets $\mathcal{F}$ on level 0



(b) $\widetilde{\mathcal{F}}$ after 1st subdivision



(c) $\widetilde{\mathcal{F}}$ after 2nd subdivision



(d) $\widetilde{\mathcal{F}}$ after 3rd subdivision

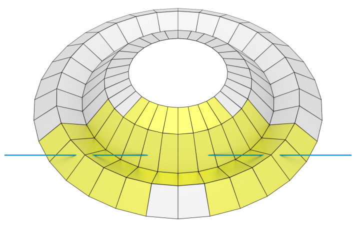
Fig. 3.15 Subdivision steps for refined detection and intersected refined facets.

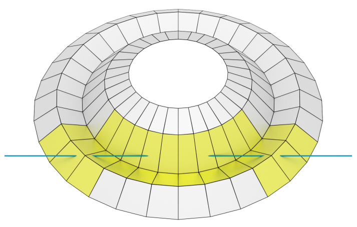
Once the k-dop bounding volume tree of the subdivision surface has been constructed, the intersection detection is performed by traversing the tree in a breadth-first manner to check if the line intersects with any node in the tree. If a node is intersected with the line, the children of the node are further checked subsequently by means of k-dops. This process proceeds until the leaf level of the tree is reached. Finally, all the potential intersected facets in the control mesh are found for further refined detection. These facets detected with the k-dop bounding volume tree are denoted with $\mathcal{F}$.

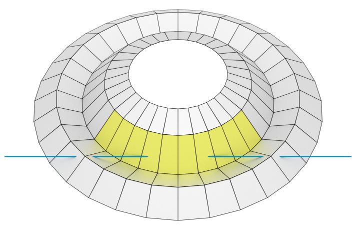
3.2.2 Detection refinement

The subdivision surface patch is not tight enough when used for the intersection detection, resulting in redundancy in $\mathcal{F}$ as shown in Figure 3.15a. In other words, there are some facets found in the hierarchical tree whose limit surfaces actually do not intersect with the line, though their defining patches do intersect with the line.

In order to reduce the redundancy in $\mathcal{F}$, a few subdivision steps are performed for the patches represented with facets in $\mathcal{F}$ and their 1-rings, as shown in Figure 3.15. After each subdivision step, the refined facets are checked again for possible intersections. The intersecting refined facets are denoted with $\widetilde{\mathcal{F}}$. Any facet in $\mathcal{F}$ which does not have an intersecting child facet is removed from $\mathcal{F}$ (Figure 3.16). The subdivision process terminates when the number of facets in $\mathcal{F}$ stops reducing.


(a) Facets $\mathcal{F}$ on level 0


(b) $\mathcal{F}$ after 1st subdivision


(c) $\mathcal{F}$ after 2nd subdivision

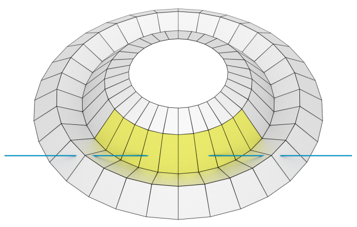
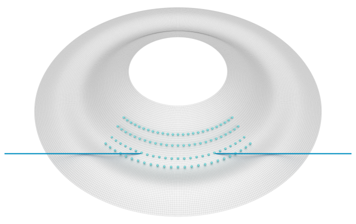

(d) $\mathcal{F}$ after 3rd subdivision

Fig. 3.16 Reduction of facets $\mathcal{F}$ on the subdivision level 0.

The remaining facets in $\mathcal{F}$ are used for computing the line/subdivision surface intersection. All the facets with regular patches can be converted to equivalent Bézier surfaces as described in Section 2.3. For facets containing EVs, approximated Bézier surfaces can be obtained by using the least-square fitting method [90]. Therefore, the intersection computation can be performed between the line and Bézier surfaces, as shown in Figure 3.17. The intersection computation between a line and a Bézier surface will be introduced in the next Section 3.2.3. Alternatively, the divide-and-conquer method described in Section 3.1.1 is applied separately for those facets with EVs.


(a) Bézier surface conversion

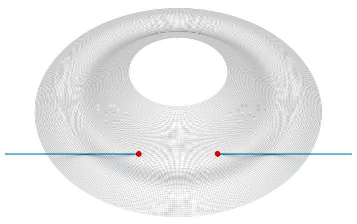

(b) Intersection points

Fig. 3.17 Conversion to Bézier surfaces and intersection of a line with Bézier surfaces.

3.2.3 Matrix-based intersection computation

Intersection algorithm

Consider the intersection of a line

$$\boldsymbol{r}(\xi) = \boldsymbol{a}\xi + \boldsymbol{b} \tag{3.59}$$

with the surface $\boldsymbol{f}(\boldsymbol{\theta})$. The line is parametrised with the scalar parameter ξ. Assuming that the line intersects the surface $\boldsymbol{f}(\boldsymbol{\theta})$ at a parameter value ξ^*, equation (3.43) has to be satisfied,

$$l^{(i)}(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{g}^{(i)}(\boldsymbol{\theta}) \cdot \begin{pmatrix} \boldsymbol{a}\xi^* + \boldsymbol{b} \\ 1 \end{pmatrix} = \sum_{\boldsymbol{j}} \widetilde{B}^{\boldsymbol{\nu}}_{\boldsymbol{j}}(\boldsymbol{\theta}) \boldsymbol{\gamma}^{(i)}_{\boldsymbol{j}} \cdot \begin{pmatrix} \boldsymbol{a}\xi^* + \boldsymbol{b} \\ 1 \end{pmatrix} = 0\,, \tag{3.60}$$

or expressed more compactly with a matrix $\boldsymbol{M}(\xi^*)$

$$\sum_{\boldsymbol{j}} \widetilde{B}^{\boldsymbol{\nu}}_{\boldsymbol{j}}(\boldsymbol{\theta}) M^{(i)}_{\boldsymbol{j}}(\xi^*) = \boldsymbol{0}\,, \tag{3.61}$$

where $M^{(i)}_{\boldsymbol{j}}$ constitutes the i-th column of the implicit M-Rep $\boldsymbol{M}$ as described in (3.45). This corresponds to the drop of the rank of the matrix $\boldsymbol{M}(\xi)$ when the surface is intersected at the line parameter ξ^*. In other words, the rank of the matrix $\boldsymbol{M}(\xi)$ drops only at the parameter values corresponding to intersection points.

The implicit M-Rep $\boldsymbol{M}(\xi)$ contains entries which are either constant or linear in ξ, i.e.,

$$\boldsymbol{M}(\xi) = \boldsymbol{A}^c - \xi \boldsymbol{B}^c\,. \tag{3.62}$$

where $\boldsymbol{A}^c$ and $\boldsymbol{B}^c$ are constant matrices. Considering that at the intersection points (3.61) is satisfied, the generalised eigenvalue problem

$$(\boldsymbol{A}^c - \xi \boldsymbol{B}^c)\boldsymbol{\phi} = \boldsymbol{0} \tag{3.63}$$

gives all the intersection points of the line with the surface, where only the non-complex eigenvalues are relevant. The eigenvectors $\boldsymbol{\phi}$ are of no significance for the intersection computations. The matrix $\boldsymbol{M}(\xi)$, and hence the matrices $\boldsymbol{A}^c$ and $\boldsymbol{B}^c$, are usually not square. To compute the generalised eigenvalues of (3.63), it is first transformed into a block column-echelon form so that the eigenvalues are only related to the diagonal blocks. By choosing orthogonal transformations which lead to square diagonal blocks, it becomes straightforward to compute the eigenvalues.

Following the Algorithm 4.1 given in [91], first of all, the SVD of the matrix $\boldsymbol{B}^c$ is computed

$$\boldsymbol{B}^c = \boldsymbol{U}_b \boldsymbol{\Sigma}_b \boldsymbol{V}_b^{\mathsf{T}} . \tag{3.64}$$

The right singular vectors are used to partition the matrices $\boldsymbol{A}^c$ and $\boldsymbol{B}^c$ such that

$$\boldsymbol{B}^c \boldsymbol{V}_b = \begin{pmatrix} \boldsymbol{B}_{11}^{c\prime} & \boldsymbol{0} \end{pmatrix} \quad \text{and} \quad \boldsymbol{A}^c \boldsymbol{V}_b = \begin{pmatrix} \boldsymbol{A}_{11}^{c\prime} & \boldsymbol{A}_{12}^{c\prime} \end{pmatrix} , \tag{3.65}$$

where the number of columns of $\boldsymbol{B}_{11}^{c\prime}$ corresponds to the rank of $\boldsymbol{B}^c$, and $\boldsymbol{0}$ denotes a zero matrix. $\boldsymbol{A}_{11}^{c\prime}$ is chosen to have the same number of columns as $\boldsymbol{B}_{11}^{c\prime}$. Next, the SVD of the matrix $\boldsymbol{A}_{12}^{c\prime}$ is computed

$$\boldsymbol{A}_{12}^{c\prime} = \boldsymbol{U}_{a_{12}} \boldsymbol{\Sigma}_{a_{12}} \boldsymbol{V}_{a_{12}}^{\mathsf{T}} , \tag{3.66}$$

and the corresponding left singular vectors are used to further partition the matrices (3.65) such that

$$\boldsymbol{U}_{a_{12}}^{\mathsf{T}} \boldsymbol{B}^c \boldsymbol{V}_b = \begin{pmatrix} \boldsymbol{B}_{11}^{c\prime\prime} & \boldsymbol{0} \\ \boldsymbol{B}_{21}^{c\prime\prime} & \boldsymbol{0} \end{pmatrix} \quad \text{and} \quad \boldsymbol{U}_{a_{12}}^{\mathsf{T}} \boldsymbol{A}^c \boldsymbol{V}_b = \begin{pmatrix} \boldsymbol{A}_{11}^{c\prime\prime} & \boldsymbol{A}_{12}^{c\prime\prime} \\ \boldsymbol{A}_{21}^{c\prime\prime} & \boldsymbol{0} \end{pmatrix} . \tag{3.67}$$

With these partitioned matrices the generalised eigenvalue problem (3.63) can be rewritten as

$$\boldsymbol{U}_{a_{12}}^{\mathsf{T}} (\boldsymbol{A}^c - \xi \boldsymbol{B}^c) \boldsymbol{V}_b \boldsymbol{\phi} = \begin{pmatrix} \boldsymbol{A}_{11}^{c\prime\prime} - \xi \boldsymbol{B}_{11}^{c\prime\prime} & \boldsymbol{A}_{12}^{c\prime\prime} \\ \boldsymbol{A}_{21}^{c\prime\prime} - \xi \boldsymbol{B}_{21}^{c\prime\prime} & \boldsymbol{0} \end{pmatrix} \boldsymbol{\phi} = \boldsymbol{0} . \tag{3.68}$$

Permuting the rows yields the desired column echelon form

$$\begin{pmatrix} \boldsymbol{A}_{21}^{c\prime\prime} - \xi \boldsymbol{B}_{21}^{c\prime\prime} & \boldsymbol{0} \\ \boldsymbol{A}_{11}^{c\prime\prime} - \xi \boldsymbol{B}_{11}^{c\prime\prime} & \boldsymbol{A}_{12}^{c\prime\prime} \end{pmatrix} \boldsymbol{\phi} = \boldsymbol{0} . \tag{3.69}$$

The eigenvalues ξ for the generalised eigenvalue problem (3.69) of the block matrix $\boldsymbol{A}_{21}^{c\prime\prime} - \xi \boldsymbol{B}_{21}^{c\prime\prime}$ in the top diagonal are the same as the ones for (3.63) because only orthogonal transformations have been applied. The top diagonal may not yet be a square matrix in which case the above sketched transformations need to be repeated.

The eigenvalues ξ give the intersection parameter values denoted earlier with ξ^* which give the intersection points $\boldsymbol{r}(\xi^*)$ by inserting them into the line equation (3.59). The corresponding parameter values $\boldsymbol{\theta}^*$ for the Bézier patch are obtained from (3.60). According (3.60) the vector of the basis function values $\widetilde{B}_{\boldsymbol{j}}^{\boldsymbol{\nu}}(\boldsymbol{\theta}^*)$ at the intersection point must be in the null space of $\boldsymbol{M}(\xi^*)$. To determine $\boldsymbol{\theta}^*$ by simply considering the ratio of pairs of Bézier basis function values it is possible to obtain linear equations that can be trivially solved, which is the inverse property described in Section 3.1.3.

Examples

A linear Bézier curve Consider a linear Bézier curve $\boldsymbol{f}(\theta)$ defined with control points $\boldsymbol{x}_1 = (0, -1, 0)$ and $\boldsymbol{x}_2 = (1, 1, 0)$. The weights w_1 and w_2 associated with the control points are assumed to be equal to one. The homogeneous coordinates of the linear Bézier curve are

$$\boldsymbol{f}(\theta) = (1-\theta)\begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} + \theta \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \theta \\ 2\theta - 1 \\ 0 \\ 1 \end{pmatrix} . \tag{3.70}$$

For a linear Bézier curve $\boldsymbol{f}(\theta)$ the smallest possible polynomial degree for the auxiliary vector $\boldsymbol{g}(\theta)$ is zero. Hence, the intersection is first computed with a constant auxiliary vector $\boldsymbol{g}(\theta)$. As discussed in Section 3.1.3 about implicit matrix representations, choosing a high-order polynomial $\boldsymbol{g}(\theta)$ is possible. Increasing the polynomial degree leads, however, to larger matrices and make the intersection computations inefficient. With the sole aim of demonstrating the process of the generalised eigenvalue computation, the intersection is then computed with a linear auxiliary vector $\boldsymbol{g}(\theta)$.

Constant auxiliary vector

The constant auxiliary vector to compute the implicit matrix representation of the linear Bézier curve (3.70) is given by

$$\boldsymbol{g}(\theta) = \boldsymbol{\gamma}_1 = \begin{pmatrix} \gamma_1^1 & \gamma_1^2 & \gamma_1^3 & \gamma_1^4 \end{pmatrix}^{\mathsf{T}} . \tag{3.71}$$

According to the orthogonality requirement (3.38), i.e. $\boldsymbol{f}(\theta) \cdot \boldsymbol{g}(\theta) = 0$, the two functions have to satisfy

$$(1-\theta)(-\gamma_1^2 + \gamma_1^4) + \theta(\gamma_1^1 + \gamma_1^2 + \gamma_1^4) = 0 , \tag{3.72}$$

or expressed in matrix form

$$\begin{pmatrix} 1-\theta & \theta \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \gamma_1^1 \\ \gamma_1^2 \\ \gamma_1^3 \\ \gamma_1^4 \end{pmatrix} = 0 . \tag{3.73}$$

By inspection, the left null space of the 2×4 matrix, denoted with $\boldsymbol{C}$ in Section 3.1.3, is spanned by the two vectors

$$\boldsymbol{\gamma}_1^{(1)} = \begin{pmatrix} -2 & 1 & 0 & 1 \end{pmatrix}^\mathsf{T} \quad \text{and} \quad \boldsymbol{\gamma}_1^{(2)} = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}^\mathsf{T} . \tag{3.74}$$

These two left null vectors can also be readily obtained with a singular value decomposition or by bringing the matrix in a reduced row echelon form, see e.g. [92]. Introducing the null vectors into (3.71) yields the two auxiliary vectors

$$\boldsymbol{g}^{(1)}(\theta) = \begin{pmatrix} -2 & 1 & 0 & 1 \end{pmatrix}^\mathsf{T} \quad \text{and} \quad \boldsymbol{g}^{(2)}(\theta) = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}^\mathsf{T} , \tag{3.75}$$

which yields, according to (3.49), the following implicitisation matrix representation

$$\boldsymbol{M}(\boldsymbol{x}) = \begin{pmatrix} -2x^1 + x^2 + 1 & x^3 \end{pmatrix} . \tag{3.76}$$

As an example, consider the intersection of $\boldsymbol{f}(\theta)$ with a second line

$$\boldsymbol{r}(\xi) = (1-\xi) \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \xi \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \xi \\ 1-\xi \\ 0 \end{pmatrix} . \tag{3.77}$$

Substituting this line equation into the implicit matrix representation (3.76) gives

$$\boldsymbol{M}(\xi) = \begin{pmatrix} -3\xi + 2 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 \end{pmatrix} + \xi \begin{pmatrix} -3 & 0 \end{pmatrix} \tag{3.78}$$

At the intersection point ξ^* the matrix $\boldsymbol{M}(\xi)$ must be rank deficient, which is trivially the case when $\xi^* = 2/3$. Inserting ξ^* in (3.77) gives the intersection point $\boldsymbol{r}(\xi^*) = (2/3,\ 1/3,\ 0)$.

Linear auxiliary vector

Next, the implicit matrix representation of the linear Bézier curve (3.70) is determined with the linear auxiliary vector

$$\boldsymbol{g}(\theta) = (1-\theta)\boldsymbol{\gamma}_1 + \theta\boldsymbol{\gamma}_2 , \tag{3.79}$$

where $\boldsymbol{\gamma}_i = (\gamma_i^1, \gamma_i^2, \gamma_i^3, \gamma_i^4)^\mathsf{T}$ with $i \in \{1,\ 2\}$. The orthogonality requirement (3.38), i.e. $\boldsymbol{f}(\theta) \cdot \boldsymbol{g}(\theta) = 0$, yields

$$(1-\theta)^2(-\gamma_1^2 + \gamma_1^4) + \theta(1-\theta)(\gamma_1^1 + \gamma_1^2 - \gamma_2^2 + \gamma_1^4 + \gamma_2^4) + \theta^2(\gamma_2^1 + \gamma_2^2 + \gamma_2^4) = 0 , \tag{3.80}$$

or in matrix form

$$\begin{pmatrix}(1-\theta)^2 & 2\theta(1-\theta) & \theta^2\end{pmatrix} \boldsymbol{C}\boldsymbol{\gamma} = 0 \tag{3.81}$$

with

$$\boldsymbol{C} = \begin{pmatrix} 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\gamma} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} .$$

It is worth pointing out that the single orthogonality requirement (3.38), as can be inferred from (3.80), yields three equations for determining the eight components of the coefficients $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$, which implies $\dim(\ker \boldsymbol{C}) = 5$. The null space of $\boldsymbol{C}$, obtained by bringing in row echelon form, is spanned by the five vectors

$$\begin{aligned} \boldsymbol{\gamma}^{(1)} &= \begin{pmatrix}-1 & 0 & 0 & 0 & -1 & 0 & 0 & 1\end{pmatrix}^\mathsf{T} , & \boldsymbol{\gamma}^{(2)} &= \begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{pmatrix}^\mathsf{T} , \\ \boldsymbol{\gamma}^{(3)} &= \begin{pmatrix}1 & 0 & 0 & 0 & -1 & 1 & 0 & 0\end{pmatrix}^\mathsf{T} , & \boldsymbol{\gamma}^{(4)} &= \begin{pmatrix}-2 & 1 & 0 & 1 & 0 & 0 & 0 & 0\end{pmatrix}^\mathsf{T} , \\ \boldsymbol{\gamma}^{(5)} &= \begin{pmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\end{pmatrix}^\mathsf{T} . \end{aligned} \tag{3.82}$$

The corresponding five auxiliary vectors follow from (3.79) with

$$\begin{aligned} \boldsymbol{g}^{(1)}(\theta) &= (-1,\, 0,\, 0,\, \theta) , & \boldsymbol{g}^{(2)}(\theta) &= (0,\, 0,\, \theta,\, 0) , \\ \boldsymbol{g}^{(3)}(\theta) &= (1-2\theta,\, \theta,\, 0,\, 0) , & \boldsymbol{g}^{(4)}(\theta) &= (2\theta-2,\, 1-\theta,\, 0,\, 1-\theta) , \\ \boldsymbol{g}^{(5)}(\theta) &= (0,\, 0,\, 1-\theta,\, 0) . \end{aligned} \tag{3.83}$$

As illustrated in Figure 3.18, the five vectors $\boldsymbol{g}^{(i)}(\theta)$ can be associated for a given θ^* value with five implicitly defined planes

$$l^{(i)}(\theta^*, \boldsymbol{x}) = \boldsymbol{g}^{(i)}(\theta^*) \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} = 0 \quad \text{with } i \in \{1, \ldots, 5\} . \tag{3.84}$$

Only at their intersection point $\boldsymbol{x}^*$ all the plane equations $l^{(i)}(\theta^*, \boldsymbol{x}^*) = 0$ are simultaneously satisfied. Due to the orthogonality condition $\boldsymbol{f}(\theta^*) \cdot \boldsymbol{g}^{(i)}(\theta^*) = 0$ the intersection point $\boldsymbol{x}^*$ must also be the surface point

$$\boldsymbol{f}(\theta^*) = \begin{pmatrix} \boldsymbol{x}(\theta^*) \\ 1 \end{pmatrix} . \tag{3.85}$$

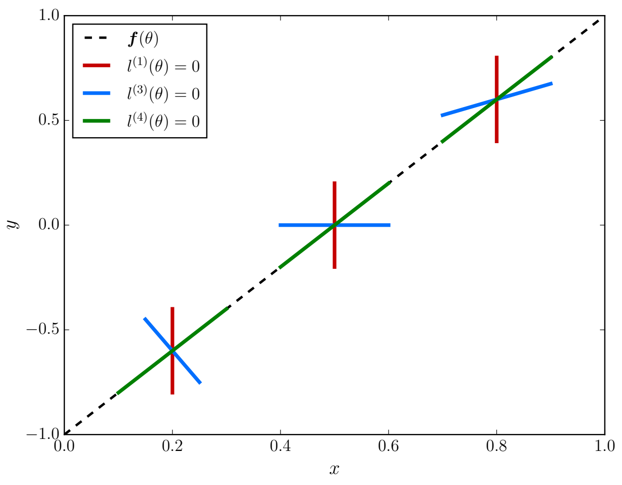


Fig. 3.18 The linear Bézier curve $\boldsymbol{f}(\theta)$ and the moving planes $l^{(i)}(\theta) = 0$ for $\theta \in \{0.2,\, 0.5,\, 0.8\}$.

The implicit matrix representation of $\boldsymbol{f}(\theta)$ is obtained according to (3.49) as follows

$$\begin{pmatrix} \boldsymbol{g}^{(1)}(\theta) \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} & \boldsymbol{g}^{(2)}(\theta) \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} & \dots & \boldsymbol{g}^{(5)}(\theta) \cdot \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1-\theta & \theta \end{pmatrix} \boldsymbol{M} \tag{3.86}$$

with the implicitisation matrix

$$\boldsymbol{M} = \begin{pmatrix} -x^1 & 0 & x^1 & -2x^1 + x^2 + 1 & x^3 \\ -x^1 + 1 & x^3 & -x^1 + x^2 & 0 & 0 \end{pmatrix} . \tag{3.87}$$

For a given θ^* and corresponding $\boldsymbol{g}(\theta^*)$ and $\boldsymbol{f}(\theta^*)$ the orthogonality condition $\boldsymbol{f}(\theta^*) \cdot \boldsymbol{g}^{(i)}(\theta^*) = 0$ gives

$$\begin{pmatrix} 1-\theta^* & \theta^* \end{pmatrix} \boldsymbol{M}(\theta^*) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix} . \tag{3.88}$$

Hence, $\boldsymbol{M}(\theta^*)$ is rank deficient and the vector $\begin{pmatrix} 1-\theta^* & \theta^* \end{pmatrix}$ lies in its left null space. Indeed, at the point $\boldsymbol{f}(\theta^*)$ the matrix has $\operatorname{rank}(\boldsymbol{M}) = 1$ and everywhere else it has $\operatorname{rank}(\boldsymbol{M}) = 2$.

As an example, consider now the intersection of $\boldsymbol{f}(\theta)$ with the line $\boldsymbol{r}(\xi)$ given in (3.77). Substituting $\boldsymbol{r}(\xi)$ into the implicitisation matrix gives

$$\boldsymbol{M}(\xi) = \begin{pmatrix} -\xi & 0 & \xi & 2-3\xi & 0 \\ -\xi+1 & 0 & 1-2\xi & 0 & 0 \end{pmatrix} = \boldsymbol{A} - \xi \boldsymbol{B} \tag{3.89}$$

with

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{B} = \begin{pmatrix} 1 & 0 & -1 & 3 & 0 \\ 1 & 0 & 2 & 0 & 0 \end{pmatrix} .$$

The generalised eigenvalue problem (3.63), i.e. $(\boldsymbol{A} - \xi\boldsymbol{B})\boldsymbol{\phi} = \boldsymbol{0}$, for computing the intersection point ξ^* is solved by applying a sequence of orthogonal transformations as discussed in Section 3.1.3. The first orthogonal transformation according to (3.65) gives the following matrices

$$\boldsymbol{A}' = \begin{pmatrix} -1.772680524 & 0.4369634904 & 0 & 0.8164965809 & 0 \\ 0.1438332239 & 1.346345178 & 0 & -0.4082482905 & 0 \end{pmatrix} , \tag{3.90}$$

$$\boldsymbol{B}' = \begin{pmatrix} -3.297858703 & 0.3523180065 & 0 & 0 & 0 \\ 0.5351687938 & 2.171081381 & 0 & 0 & 0 \end{pmatrix} . \tag{3.91}$$

After applying two subsequent steps of orthogonal transformations $\boldsymbol{A}$ and $\boldsymbol{B}$ are reduced to,

$$\boldsymbol{A}''' = \begin{pmatrix} 1.549193338 \end{pmatrix} \quad \text{and} \quad \boldsymbol{B}''' = \begin{pmatrix} 2.323790008 \end{pmatrix} . \tag{3.92}$$

The generalised eigenvalue of matrices $\boldsymbol{A}'''$ and $\boldsymbol{B}'''$ is trivially $\xi^* = 0.666666666$ and the coordinate of the intersection point is $\boldsymbol{r}(\xi^*) = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} & 0 \end{pmatrix}^{\mathsf{T}}$.

At the intersection point, the implicitisation matrix becomes

$$\boldsymbol{M}(\xi^*) = \begin{pmatrix} -\frac{2}{3} & 0 & \frac{2}{3} & 0 & 0 \\ \frac{1}{3} & 0 & -\frac{1}{3} & 0 & 0 \end{pmatrix} \tag{3.93}$$

with its left null space spanned by the vector $\begin{pmatrix} \frac{1}{2} & 1 \end{pmatrix}$. The parameter θ^* of the intersection point can be readily computed according to (3.86) since the vector $\begin{pmatrix} 1-\theta^* & \theta^* \end{pmatrix}$ has to be collinear to $\begin{pmatrix} \frac{1}{2} & 1 \end{pmatrix}$. Hence, $\theta^* = \frac{2}{3}$.

A cubic Bézier curve The same cubic Bézier curve as shown in Figure 3.10a is considered. The implicit matrix representation of the curve is obtained as (3.55). The intersections with three lines were computed. The three lines considered are plotted in Figure 3.19. Line $\boldsymbol{r}_1(\xi)$ is the x-axis, line $\boldsymbol{r}_2(\xi)$ is horizontal and tangential to the curve, and line $\boldsymbol{r}_3(\xi)$ is a vertical line with $x = 2$.

The same procedure with the previous linear Bézier curve example yields the intersection points with line $\boldsymbol{r}_1 = (4\xi, 0)$ as follows,

$$\xi_1 = 0, \quad \xi_2 = 0.574074, \quad \xi_3 = 1 . \tag{3.94}$$

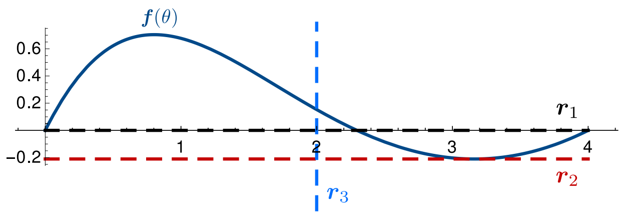


Fig. 3.19 Intersection of the cubic Bézier curve with the three lines.

It can be inferred from the first derivative of the cubic Bézier curve that a horizontal line intersects the curve tangentially at two points with the curve parameter $\theta = (5+\sqrt{7})/9$ or $\theta = (5-\sqrt{7})/9$. The line $\boldsymbol{r}_2$ is considered as intersecting tangentially with the curve at $\theta = (5+\sqrt{7})/9$. The computation with line $\boldsymbol{r}_2$ yields the following line parameters of the intersection points,

$$\xi_1 = 0.7904215, \quad \xi_2 = 0.7904215, \quad \xi_3 = -0.0243004 \tag{3.95}$$

with two of them the same value corresponding to the tangential intersection point and the other one intersecting at the point outside the domain $[0,1]$.

The computation with line $\boldsymbol{r}_3 = (2, 2\xi - 1)$ yields the following eigenvalues,

$$\xi_1 = 0.576487, \quad \xi_2 = 36.461756 + 10.897286i, \quad \xi_3 = 36.461756 - 10.897286i\,, \tag{3.96}$$

where only ξ_1 is valid which corresponds to the only intersection point intersected by the line.

3.3 Surface-fitted lattice generation

In our implementation of creating lattice-skin structures, lattices with periodic cubic cells are considered, for example, octet, pyramidal and BCC (body-centred cell). The cell size and the cell type can be prescribed by the user. Each cubic cell contains an arrangement of a small number of struts with the same size. The assumption of cubic cells is here not overly restrictive, considering that amongst polyhedra with a small number of faces only the cube and the dodecahedron are able to uniformly fill the space [93]. Lattices containing several different types of polyhedral cells or with gradually changing cell size are possible but will be not considered for the sake of simplicity.

In order to generate the periodic lattice structure, the subdivision surface is first immersed in a lattice grid as shown in Figure 3.20. The cells in the lattice grid can be

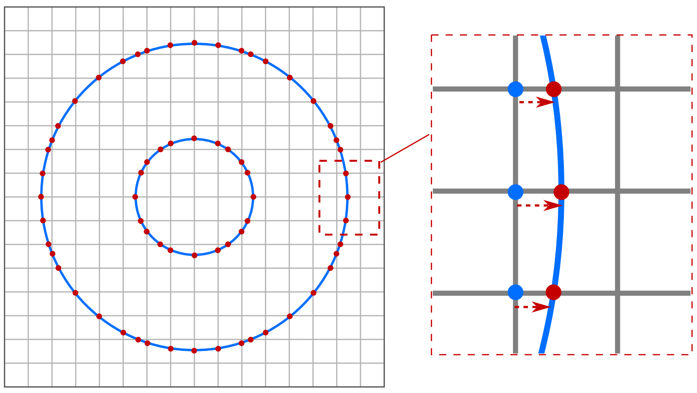


Fig. 3.20 Surface and the lattice and the projection of cell corners on to the surface (two-dimensional illustration).

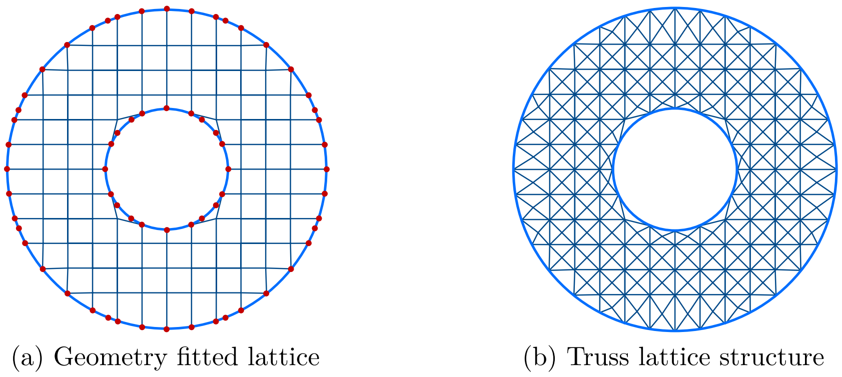


Fig. 3.21 Truss lattice generation from a geometry fitted lattice.

categorised into three types, i.e. inside cells, outside cells and cut cells. Cut cells are those with edges intersecting with the surface. Inside cells will remain, and the shapes of cut cells are modified by projecting their vertices to the surface. Outside cells will be removed before adding struts into the lattice grid to form a lattice structure. The three types of cells can be distinguished according to the intersection points of the lattice grid with the surface.

The lattice grid can be seen as a set of line segments. The intersection points of the lattice grid with the surface is therefore obtained by looping over all line segments of the lattice grid and computing the line/subdivision surface intersection repeatedly with the algorithm described in Section 3.2.3. When all the intersection points are computed, each lattice grid node can be distinguished if it is inside or outside the surface by examining their positions on the lattice line relative to the intersection points. For a closed surface, the lattice nodes between the first and second, third and fourth, and so on intersection points have to be inside the surface. If a node is coincident with an intersection point, it is regarded as an inside node. Distinguishing cell types becomes straightforward after specifying the positions of lattice grid nodes.

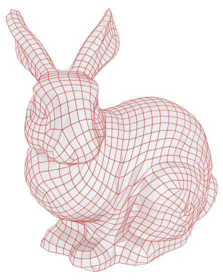
Before adding struts into the lattice grid, the nodes of cut cells are moved to fit the surface. For an edge intersecting with the surface, the end node closest to the intersection point is moved to the intersection point. A surface-fitted lattice grid is generated as shown in Figure 3.21a. After a lattice grid fitting the surface is created, it becomes straightforward to add struts in cells to form the cell type prescribed (Figure 3.21b).

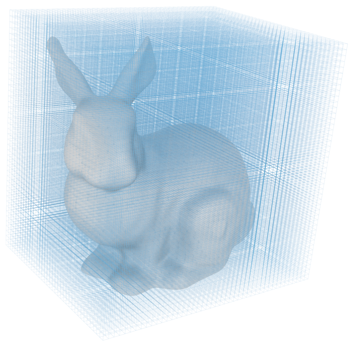
3.4 Example of the Stanford bunny

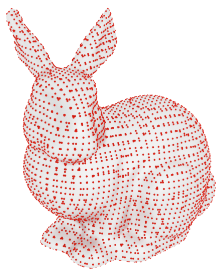
In this example the timings for generating a lattice within a given surface by using the aforementioned algorithms are investigated. Figure 3.22 shows the steps involved in generating an internal lattice for the Stanford bunny. The bunny surface is a Catmull-Clark subdivision surface and consists of 2071 facets. The lattice size is chosen with $45 \times 45 \times 37$ (the number of lattice joints in each direction). This leads to 5704 intersection points between the lattice and the surface. Each lattice cell is tessellated with struts to give a body-centred cubic unit cell connectivity as shown in Figure 3.22d.

All the intersection points between the surface and lattice are computed using the intersection algorithm developed in Section 3.2.3. The SVD computations involved are computed with the open source library Eigen [94]. For intersection detection we use 14-dops, consisting of six directions orthogonal to the coordinate planes, two directions parallel to the average normal of all the Bézier patches (evaluated by sampling), and six directions orthogonal to the lattice. In Figure 3.23 the 14-dop and the axis-aligned bounding box (with k = 6) are compared in terms of efficiency of the intersection detection with a lattice grid of the size $19 \times 17 \times 17$. Different lattice grid orientations $\{0°, 15°, 30°, 45°\}$ have been considered with respect to the bunny axis. As to be expected, using the 14-dop is far more efficient than the axis-aligned bounding box when the lattice grid is not aligned with the coordinate axis, since the bounding box size of a lattice edge depends on the lattice orientation.

For evaluating the accuracy and efficiency of the implicitisation, we compare it with the widely used subdivision method as described in Section 3.1.1. In the subdivision method the Bézier patches are successively refined until they are sufficiently flat so that the intersection computation reduces to the trivial intersection between a triangle and a line segment. The flatness criterion employed in the subdivision method is to consider the difference between the maximum support height $h_{j,\max}^{\text{patch}}$ and the minimum support


(a) Control mesh


(b) Immersed lattice


(c) Intersection points

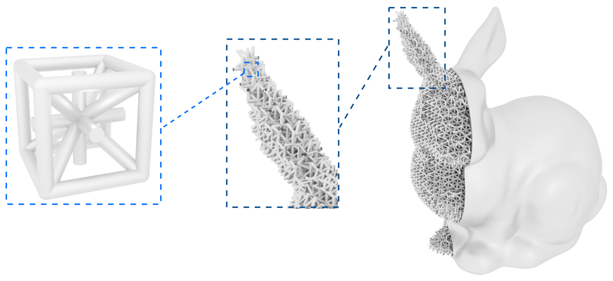
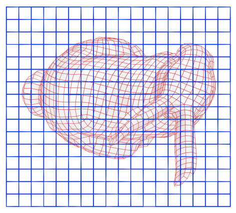
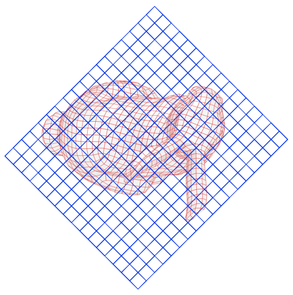
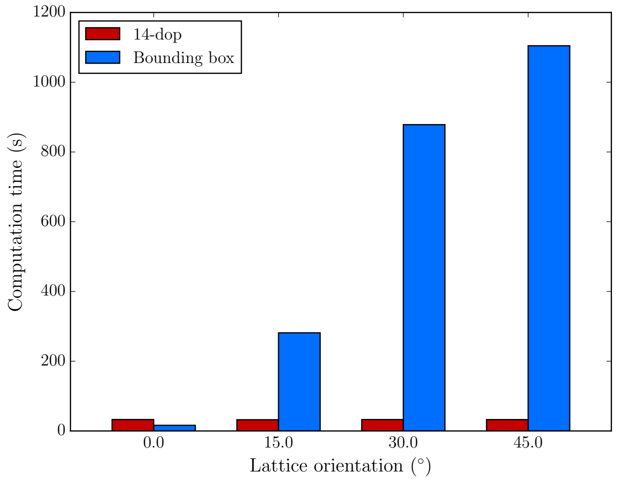

(d) Lattice structure with BCC unit cells

Fig. 3.22 Generation of a lattice-skin structure for the Stanford bunny and its deformation under a loading.


(a) Grid orientation 0°


(b) Grid orientation 45°



(c) Computation time using k-dops and bounding box

Fig. 3.23 Comparison of efficiency using k-dops and bounding box.

Table 3.1 Accuracy of intersection computation with implicitisation.

Tolerance tol_{dc}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}
Avg[log($\|\|\boldsymbol{x}_{\text{implicit}} - \boldsymbol{x}_{\text{dc}}\|\|$)]	-4.84	-6.81	-8.78	-10.81	-12.53

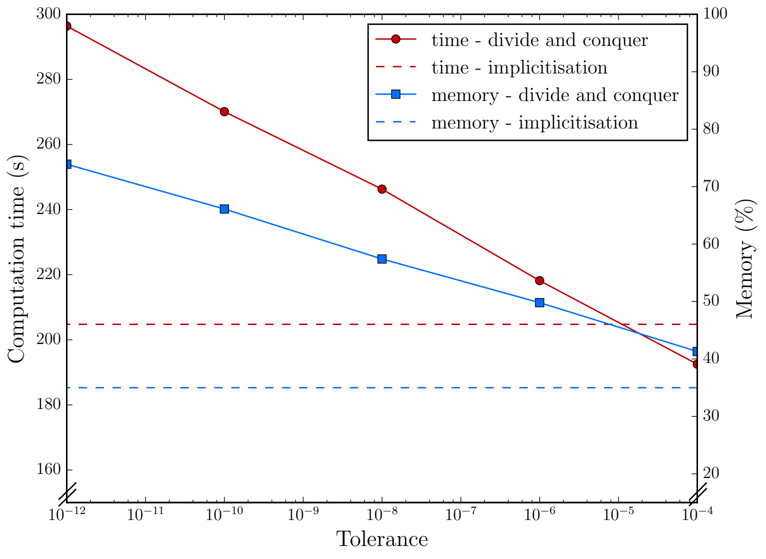


Fig. 3.24 Performance of intersection computations with the implicitisation and the divide-and-conquer method.

height $h_{j,\text{min}}^{\text{patch}}$ along the direction of the average normal of the Bézier patch (see (3.1)), and this difference has to be less than a given tolerance. The tolerance controls the accuracy of the intersection computation. Note that this tolerance becomes ineffective when the line is (nearly) tangential to the surface, which is not a trivial problem for the subdivision method [95].

The average difference between all the intersection points obtained by the implicitisation and the divide-and-conquer listed in Table 3.1 shows that the overall accuracy using implicitisation to calculate intersection points can reach at least 10^{-12}. The performance comparison between the implicitisation and the subdivision methods is shown in Figure 3.24. Both the computation time and memory consumed increase linearly with the decreasing tolerance for the subdivision method. This reflects the linear relationship between the depth of the bounding volume tree and the required accuracy. In contrast, the implicitisation exhibits a high accuracy (more than 10^{-12}) but takes less time and memory compared with the subdivision method. The computation was performed on a computer equipped with an Intel Xeon(R) CPU E5-2623 v3 @ 3.00GHz processor and 32GB memory.

Chapter 4

Analysis of lattice-skin structures

In this chapter the finite element analysis of the lattice-skin structure is demonstrated considering the lattice-skin coupling. In the isogeometric framework, the finite element model and the geometry model use the same basis functions. Since the thin-shell in the lattice-skin structure is represented with the subdivision surface, the subdivision basis functions are used in the finite element analysis of the thin-shell. A brief review of the finite element analysis of the thin-shell using subdivision basis functions is given in Section 4.1. The lattice-coupling is considered with Lagrange multipliers and the finite element discretisation of equilibrium equations of the lattice-skin structure is derived in Section 4.2. Examples including a verification example of a sandwich plate are given in Section 4.3.

4.1 Review of subdivision thin-shell finite elements

4.1.1 Energy functional of deformed thin-shells

The thin-shell is characterised with its mid-surface, and the Kirchhoff-Love theory is adopted for the thin-shell analysis. In the Kirchhoff-Love theory, the normal to the mid-surface remains normal after deformation, and there is no through-the-thickness deformation.

The mid-surface of the thin shell is parameterised with curvilinear coordinates $(\theta^1, \theta^2) \in \mathbb{R}^2$. An illustration of the mid-surface in the reference and deformed configurations is given in Figure 4.1. In the following the superscript s is introduced in order to distinguish between the variables for the thin-shell and the lattice in the lattice-skin structure.

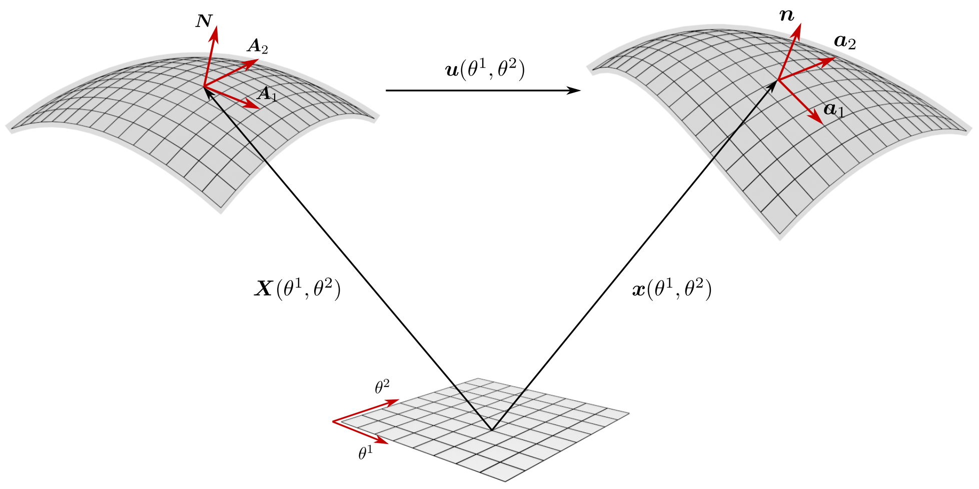


Fig. 4.1 Geometry of the thin-shell in the reference and deformed configurations.

The covariant basis vectors of the mid-surface are computed with

$$\boldsymbol{A}_\alpha = \frac{\partial \boldsymbol{X}^{\mathrm{s}}}{\partial \theta^\alpha}\,, \quad \boldsymbol{a}_\alpha = \frac{\partial \boldsymbol{x}^{\mathrm{s}}}{\partial \theta^\alpha}, \quad \alpha = 1, 2\,, \tag{4.1}$$

where $\boldsymbol{X}^{\mathrm{s}}(\theta^1,\theta^2)$ and $\boldsymbol{x}^{\mathrm{s}}(\theta^1,\theta^2)$ are position vectors of material points on the mid-surface in the reference and deformed configurations respectively, which are related by the displacement field $\boldsymbol{u}^{\mathrm{s}}(\theta^1,\theta^2)$ as

$$\boldsymbol{x}^{\mathrm{s}}(\theta^1,\theta^2) = \boldsymbol{X}^{\mathrm{s}}(\theta^1,\theta^2) + \boldsymbol{u}^{\mathrm{s}}(\theta^1,\theta^2)\,. \tag{4.2}$$

The unit normals of the mid-surface in reference and deformed configurations are

$$\boldsymbol{N} = \frac{\boldsymbol{A}_1 \times \boldsymbol{A}_2}{|\boldsymbol{A}_1 \times \boldsymbol{A}_2|}\,, \quad \boldsymbol{n} = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}\,. \tag{4.3}$$

Considering the linearisation of the Green-Lagrange strain tensor in terms of the shell thickness, the membrane strain tensor $\boldsymbol{\alpha}$ and the bending strain tensor $\boldsymbol{\beta}$ are

$$\boldsymbol{\alpha} = \alpha_{\alpha\beta}\,\boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta = \frac{1}{2}(\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta - \boldsymbol{A}_\alpha \cdot \boldsymbol{A}_\beta)\,\boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta\,, \tag{4.4}$$

$$\boldsymbol{\beta} = \beta_{\alpha\beta}\,\boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta = (\boldsymbol{a}_\alpha \cdot \boldsymbol{n}_{,\beta} - \boldsymbol{A}_\alpha \cdot \boldsymbol{N}_{,\beta})\,\boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta \tag{4.5}$$

with the comma denoting partial differentiation.

The internal potential energy of the deformed thin-shell can be computed with

$$\Pi_{\text{shell}}(\boldsymbol{u}^{\mathrm{s}}) = \int_{\Omega^{\mathrm{s}}} (W^m(\boldsymbol{\alpha}) + W^b(\boldsymbol{\beta}))\,\mathrm{d}\Omega^{\mathrm{s}}\,, \tag{4.6}$$

where the membrane strain energy density W^m and the bending strain energy density W^b are defined as

$$W^m(\boldsymbol{\alpha}) = \frac{1}{2}\frac{Et}{1-\nu^2}\boldsymbol{\alpha} : \boldsymbol{H} : \boldsymbol{\alpha}\,, \tag{4.7}$$

$$W^b(\boldsymbol{\beta}) = \frac{1}{2}\frac{Et^3}{12(1-\nu^2)}\boldsymbol{\beta} : \boldsymbol{H} : \boldsymbol{\beta}\,, \tag{4.8}$$

with E, ν and t denoting the Young's modulus, the Poisson's ratio and the thickness of the thin-shell respectively; $\boldsymbol{H}$ denoting the fourth-order elasticity tensor defined as

$$\boldsymbol{H} = \left[\nu A^{\alpha\beta}A^{\gamma\delta} + \frac{1}{2}(1-\nu)(A^{\alpha\gamma}A^{\beta\delta} + A^{\alpha\delta}A^{\beta\gamma})\right] \boldsymbol{A}_\alpha \otimes \boldsymbol{A}_\beta \otimes \boldsymbol{A}_\gamma \otimes \boldsymbol{A}_\delta \tag{4.9}$$

with the contravariant metric in the reference domain $A^{\alpha\beta} = \boldsymbol{A}^\alpha \cdot \boldsymbol{A}^\beta$.

4.1.2 Finite element discretisation

The total potential energy of the thin-shell is

$$\Pi^{\mathrm{s}}(\boldsymbol{u}^{\mathrm{s}}) = \Pi_{\mathrm{shell}}(\boldsymbol{u}^{\mathrm{s}}) - \Pi_{\mathrm{ext}}(\boldsymbol{u}^{\mathrm{s}})\,, \tag{4.10}$$

where Π_{ext} is the work done by external loads,

$$\Pi_{\mathrm{ext}} = \int_{\Omega^{\mathrm{s}}} \boldsymbol{q}\cdot\boldsymbol{u}^{\mathrm{s}}\,\mathrm{d}\Omega^{\mathrm{s}} + \int_{\Gamma^{\mathrm{s}}} \boldsymbol{\tau}\cdot\boldsymbol{u}^{\mathrm{s}}\,\mathrm{d}\Gamma^{\mathrm{s}} \tag{4.11}$$

with $\boldsymbol{q}$ denoting the distributed loading on the mid-surface and $\boldsymbol{\tau}$ the force applied along the boundary of the thin-shell.

The equilibrium equation can be obtained by computing the stationary point of the potential energy functional (4.10), where the first variation has to diminish,

$$\begin{aligned}\frac{\partial\Pi^{\mathrm{s}}(\boldsymbol{u})}{\partial\boldsymbol{u}^{\mathrm{s}}}\delta\boldsymbol{u}^{\mathrm{s}} &= \frac{\partial\Pi_{\mathrm{shell}}(\boldsymbol{u}^{\mathrm{s}})}{\partial\boldsymbol{u}^{\mathrm{s}}}\delta\boldsymbol{u}^{\mathrm{s}} - \frac{\partial\Pi_{\mathrm{ext}}(\boldsymbol{u}^{\mathrm{s}})}{\partial\boldsymbol{u}^{\mathrm{s}}}\delta\boldsymbol{u}^{\mathrm{s}} \\ &= \int_{\Omega^{\mathrm{s}}}\left(\frac{\partial W^m(\boldsymbol{\alpha})}{\partial\boldsymbol{u}^{\mathrm{s}}}\delta\boldsymbol{u}^{\mathrm{s}} + \frac{\partial W^b(\boldsymbol{\beta})}{\partial\boldsymbol{u}^{\mathrm{s}}}\delta\boldsymbol{u}^{\mathrm{s}}\right)\mathrm{d}\Omega^{\mathrm{s}} - \int_{\Omega^{\mathrm{s}}}\boldsymbol{q}\cdot\delta\boldsymbol{u}^{\mathrm{s}}\,\mathrm{d}\Omega^{\mathrm{s}} - \int_{\Gamma^{\mathrm{s}}}\boldsymbol{\tau}\cdot\delta\boldsymbol{u}^{\mathrm{s}}\,\mathrm{d}\Gamma^{\mathrm{s}} \\ &= 0\,.\end{aligned} \tag{4.12}$$

Subdivision basis functions [13] are used for the finite element discretisation of the weak form (4.12) such that the position $\boldsymbol{x}^{\mathrm{s}}$ and the displacement field $\boldsymbol{u}^{\mathrm{s}}$ of the mid-surface are approximated as $\mathbf{x}^{\mathrm{s}}_h$ and $\mathbf{u}^{\mathrm{s}}_h$ respectively,

$$\mathbf{x}^{\mathrm{s}}_h(\theta^1, \theta^2) = \sum_{i=1}^{n_s} N^{\mathrm{s}}_i(\theta^1, \theta^2)\mathbf{x}^{\mathrm{s}}_i\,, \tag{4.13}$$

$$\mathbf{u}^{\mathrm{s}}_h(\theta^1, \theta^2) = \sum_{i=1}^{n_s} N^{\mathrm{s}}_i(\theta^1, \theta^2)\mathbf{u}^{\mathrm{s}}_i\,, \tag{4.14}$$

where $\mathbf{x}^{\mathrm{s}}_i$ and $\mathbf{u}^{\mathrm{s}}_i$ are nodal positions and displacements of finite elements, N^{s}_i are associated subdivision basis functions, and n_s denotes the number of support nodes in an element. In the following context, the discretised fields are denoted without the subscript h.

In the case of small displacement of the mid-surface, the linearised membrane strain $\boldsymbol{\alpha}_{\mathrm{lin}}$ and the linearised bending strain $\boldsymbol{\beta}_{\mathrm{lin}}$ are given by, according to [13],

$$\boldsymbol{\alpha}_{\mathrm{lin}} = \frac{1}{2}\left(\boldsymbol{A}_\alpha \cdot \boldsymbol{u}_{,\beta} + \boldsymbol{u}_{,\alpha} \cdot \boldsymbol{A}_\beta\right) \boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta\,, \tag{4.15}$$

$$\begin{aligned}\boldsymbol{\beta}_{\mathrm{lin}} = &-\boldsymbol{u}_{,\alpha\beta} \cdot \boldsymbol{N} + \frac{1}{|\mathcal{J}|}\left[\boldsymbol{u}_{,1} \cdot (\boldsymbol{A}_{\alpha,\beta} \times \boldsymbol{A}_2) + \boldsymbol{u}_{,2} \cdot (\boldsymbol{A}_1 \times \boldsymbol{A}_{\alpha,\beta})\right] \\ &+ \frac{\boldsymbol{N} \cdot \boldsymbol{A}_{\alpha,\beta}}{|\mathcal{J}|}\left[\boldsymbol{u}_{,1} \cdot (\boldsymbol{A}_2 \times \boldsymbol{N}) + \boldsymbol{u}_{,2} \cdot (\boldsymbol{N} \times \boldsymbol{A}_1)\right] \boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta\end{aligned} \tag{4.16}$$

with the Jacobian of the mid-surface $|\mathcal{J}| = |\boldsymbol{A}_1 \times \boldsymbol{A}_2|$. Considering the discretised displacement field (4.14) of the mid-surface, the linearised membrane strain and the bending strain can be approximated as

$$\boldsymbol{\alpha}_{\mathrm{lin}}(\theta^1, \theta^2) = \sum_{i=1}^{n_s} \mathbf{M}^{\mathrm{s}}_i(\theta^1, \theta^2)\mathbf{u}^{\mathrm{s}}_i\,, \tag{4.17}$$

$$\boldsymbol{\beta}_{\mathrm{lin}}(\theta^1, \theta^2) = \sum_{i=1}^{n_s} \mathbf{B}^{\mathrm{s}}_i(\theta^1, \theta^2)\mathbf{u}^{\mathrm{s}}_i \tag{4.18}$$

with

$$[\mathbf{M}^{\mathrm{s}}_i]_{\alpha\beta} = \frac{1}{2}\left(\boldsymbol{A}_\alpha \cdot N^{\mathrm{s}}_{i,\beta} + N^{\mathrm{s}}_{i,\alpha} \cdot \boldsymbol{A}_\beta\right) \tag{4.19}$$

and

$$\begin{aligned}[\mathbf{B}^{\mathrm{s}}_i]_{\alpha\beta} = &-N^{\mathrm{s}}_{i,\alpha\beta} \cdot \boldsymbol{N} + \frac{1}{|\mathcal{J}|}\left[N^{\mathrm{s}}_{i,1} \cdot (\boldsymbol{A}_{\alpha,\beta} \times \boldsymbol{A}_2) + N^{\mathrm{s}}_{i,2} \cdot (\boldsymbol{A}_1 \times \boldsymbol{A}_{\alpha,\beta})\right] \\ &+ \frac{\boldsymbol{N} \cdot \boldsymbol{A}_{\alpha,\beta}}{|\mathcal{J}|}\left[N^{\mathrm{s}}_{i,1} \cdot (\boldsymbol{A}_2 \times \boldsymbol{N}) + N^{\mathrm{s}}_{i,2} \cdot (\boldsymbol{N} \times \boldsymbol{A}_1)\right]\,.\end{aligned} \tag{4.20}$$

Finally, the stiffness matrix of the discretised equilibrium equation is computed with

$$\mathsf{K}^{\mathrm{s}}_{ij} = \sum_e \int_{\Omega^{\mathrm{s}}_e} \left[\frac{Et}{1-\nu^2} \mathsf{M}^{\mathrm{s}\,\mathsf{T}}_i \mathsf{H} \mathsf{M}^{\mathrm{s}}_j + \frac{Et^3}{12(1-\nu^2)} \mathsf{B}^{\mathrm{s}\,\mathsf{T}}_i \mathsf{H} \mathsf{B}^{\mathrm{s}}_j \right] \mathrm{d}\Omega^{\mathrm{s}}_e \,. \tag{4.21}$$

Under the subdivision scheme the Dirichlet boundary conditions can be applied considering the fact that it is essentially a cubic B-spline along the edge in the Catmull-Clark subdivision and the Loop subdivision. Therefore, a cubic B-spline boundary can be constructed along the boundary by considering ghost vertices outside the domain [13]. Alternatively, as mentioned in Section 2.2.3 the extended subdivision schemes are derived based on the cubic B-spline boundary conditions, it becomes straightforward to apply the Dirichlet boundaries using the extended subdivision schemes directly [15]. The Dirichlet boundary conditions can also be imposed weakly with the Nitsche's method [96, 97], which employs work-conjugate pairs with adjoint terms in the weak form so that variational consistency and symmetry are guaranteed in the weak form formulation when compared with penalty methods [98].

4.2 Lattice-skin coupling and discretisation

The lattice in the lattice-skin structure is modelled with truss elements which do not transfer moments. Each strut deforms only by stretching without bending. This approximation is sufficient for lattices which are stretch dominated [99, 100]. It is straightforward to extend the lattice-skin coupling with truss elements to beam elements or solid elements. It is worth noting that modelling struts with Timoshenko beam elements or solid elements is better aligned with the lattice structure having struts with relatively high ratios of diameter to length (for example, greater than $1/10 \sim 1/8$ [101]). In the following context of analysis and optimisation, as the stretch-dominated lattice structure is considered, the restriction of strut dimensions is relaxed so that the strut with a relatively high ratio of diameter to length is still modelled with the truss element, and it would not affect the overall approaches and the conclusions presented.

The unit directional vector of a truss element is given by

$$\boldsymbol{t}^{\mathrm{l}}_j = \frac{\boldsymbol{X}^{\mathrm{l}}_{j2} - \boldsymbol{X}^{\mathrm{l}}_{j1}}{L_j} \,, \tag{4.22}$$

where $\boldsymbol{X}^{\mathrm{l}}_{j1}$ and $\boldsymbol{X}^{\mathrm{l}}_{j2}$ denote coordinates of the two joints of the strut indexed with j, and L_j is the length of the strut.

In the case of small displacement, the linearised stretch strain ϵ_j of the j-th strut after the deformation is

$$\epsilon_j = \frac{\boldsymbol{u}^{\mathrm{l}}_{j2} - \boldsymbol{u}^{\mathrm{l}}_{j1}}{L_j} \cdot \boldsymbol{t}^{\mathrm{l}}_j \,, \tag{4.23}$$

where $\boldsymbol{u}^{\mathrm{l}}_{j1}$ and $\boldsymbol{u}^{\mathrm{l}}_{j2}$ are joint displacements. The stretch energy density of a strut is computed with

$$W^a(\epsilon_j) = \frac{1}{2} E_j \epsilon_j^2 \tag{4.24}$$

with E_j the Young's modulus of the j-th strut. The internal potential energy of the lattice is given by

$$\Pi_{\text{lattice}}(\boldsymbol{u}^{\mathrm{l}}) = \sum_j W^a(\epsilon_j) V_j \,, \tag{4.25}$$

where $\boldsymbol{u}^{\mathrm{l}}$ is the nodal displacement vector of the lattice, and V_j is the volume of the strut.

Hence, the total potential energy of the lattice-skin coupled structure is given by

$$\Pi(\boldsymbol{u}^{\mathrm{s}}, \boldsymbol{u}^{\mathrm{l}}) = \Pi_{\text{shell}}(\boldsymbol{u}^{\mathrm{s}}) + \Pi_{\text{lattice}}(\boldsymbol{u}^{\mathrm{l}}) - \Pi_{\text{ext}} \,, \tag{4.26}$$

where $\boldsymbol{u}^{\mathrm{s}}$ is the displacement field of the thin-shell, $\boldsymbol{u}^{\mathrm{l}}$ is the displacement vector of lattice nodes, Π_{ext} is the work done by external forces including the external loading applied on lattice joints. The internal potential energy of the thin shell $\Pi_{\text{shell}}(\boldsymbol{u}^{\mathrm{s}})$ has been given in Section 4.1.1.

Some of the lattice nodes are located on the mid-surface of the thin-shell, and they are required to have compatible displacements at the coupled positions. The set of the coupled lattice nodes is denoted with $\mathcal{D}^{\mathrm{l}}_{\mathrm{c}}$, and the coupled thin-shell nodes are in the set $\mathcal{D}^{\mathrm{s}}_{\mathrm{c}}$. The coupled positions, including their physical coordinates and parametric coordinates on the mid-surface of the thin-shell, have been obtained from the intersection computation introduced in Chapter 3. The displacement compatibility condition between the thin-shell and the lattice at the positions of coupled lattice nodes is

$$\boldsymbol{u}^{\mathrm{s}}(\boldsymbol{\theta}^{\mathrm{l}}_i) = \boldsymbol{u}^{\mathrm{l}}_i \,, \quad \forall i \in \mathcal{D}^{\mathrm{l}}_{\mathrm{c}} \tag{4.27}$$

with $\boldsymbol{u}^{\mathrm{l}}_i$ the displacement vector of the i-th lattice node which is coupled with the thin-shell, and $\boldsymbol{\theta}^{\mathrm{l}}_i$ the parametric coordinates of the coupled lattice node $\boldsymbol{X}^{\mathrm{l}}_i$. This condition can be imposed with Lagrange multipliers $\boldsymbol{\lambda}_i$ by considering the Lagrangian of the total potential energy functional of the lattice-skin structure

$$L(\boldsymbol{u}^{\mathrm{s}}, \boldsymbol{u}^{\mathrm{l}}, \boldsymbol{\lambda}) = \Pi_{\text{shell}}(\boldsymbol{u}^{\mathrm{s}}) + \Pi_{\text{lattice}}(\boldsymbol{u}^{\mathrm{l}}) - \Pi_{\text{ext}}(\boldsymbol{u}^{\mathrm{s}}, \boldsymbol{u}^{\mathrm{l}}) + \sum_i \boldsymbol{\lambda}_i \left(\boldsymbol{u}^{\mathrm{s}}(\boldsymbol{\theta}^{\mathrm{l}}_i) - \boldsymbol{u}^{\mathrm{l}}_i \right) \,. \tag{4.28}$$

The equilibrium equations can be obtained by computing the stationary point of the functional (4.28) as follows,

$$\frac{\partial \Pi_{\text{shell}}(\boldsymbol{u}^{\text{s}})}{\partial \boldsymbol{u}^{\text{s}}}\delta \boldsymbol{u}^{\text{s}} - \frac{\partial \Pi_{\text{ext}}}{\partial \boldsymbol{u}^{\text{s}}}\delta \boldsymbol{u}^{\text{s}} + \sum_i \boldsymbol{\lambda}_i \delta \boldsymbol{u}^{\text{s}}(\boldsymbol{\theta}^{\text{l}}_i) = 0 \quad i \in \mathcal{D}^{\text{l}}_{\text{c}}\,, \tag{4.29}$$

$$\frac{\partial \Pi_{\text{lattice}}(\boldsymbol{u}^{\text{l}}_i)}{\partial \boldsymbol{u}^{\text{l}}_i} - \frac{\partial \Pi_{\text{ext}}}{\partial \boldsymbol{u}^{\text{l}}_i} = 0 \quad i \notin \mathcal{D}^{\text{l}}_{\text{c}}\,, \tag{4.30}$$

$$\frac{\partial \Pi_{\text{lattice}}(\boldsymbol{u}^{\text{l}}_i)}{\partial \boldsymbol{u}^{\text{l}}_i} - \frac{\partial \Pi_{\text{ext}}}{\partial \boldsymbol{u}^{\text{l}}_i} - \boldsymbol{\lambda}_i = 0 \quad i \in \mathcal{D}^{\text{l}}_{\text{c}}\,, \tag{4.31}$$

$$\boldsymbol{u}^{\text{s}}(\boldsymbol{\theta}^{\text{l}}_i) - \boldsymbol{u}^{\text{l}}_i = 0 \quad i \in \mathcal{D}^{\text{l}}_{\text{c}}\,. \tag{4.32}$$

With the displacement field of the thin-shell discretised with (4.14), the finite element discretisation of these equations yields the following discretised equilibrium equation system with subscript c denoting the coupled degrees-of-freedom and subscript d denoting the non-coupled degrees-of-freedom,

$$\begin{pmatrix} \mathbf{K}^{\text{s}}_{\text{dd}} & \mathbf{K}^{\text{s}}_{\text{dc}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}^{\text{s}}_{\text{cd}} & \mathbf{K}^{\text{s}}_{\text{cc}} & \mathbf{0} & \mathbf{0} & \mathbf{G}^{\mathsf{T}}_{\text{c}} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}^{\text{l}}_{\text{dd}} & \mathbf{K}^{\text{l}}_{\text{dc}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}^{\text{l}}_{\text{cd}} & \mathbf{K}^{\text{l}}_{\text{cc}} & -\mathbf{H}^{\mathsf{T}}_{\text{c}} \\ \mathbf{0} & \mathbf{G}_{\text{c}} & \mathbf{0} & -\mathbf{H}_{\text{c}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{\text{s}}_{\text{d}} \\ \mathbf{u}^{\text{s}}_{\text{c}} \\ \mathbf{u}^{\text{l}}_{\text{d}} \\ \mathbf{u}^{\text{l}}_{\text{c}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{\text{s}}_{\text{d}} \\ \mathbf{f}^{\text{s}}_{\text{c}} \\ \mathbf{f}^{\text{l}}_{\text{d}} \\ \mathbf{f}^{\text{l}}_{\text{c}} \\ \mathbf{0} \end{pmatrix}, \tag{4.33}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ is the displacement vector, $\mathbf{f}$ is the external force vector; $\mathbf{H}_{\text{c}}$ is an identity matrix, i.e. $\mathbf{H}_{\text{c}} = \mathbf{I}$, and the matrix $\mathbf{G}_{\text{c}}$ takes the following form

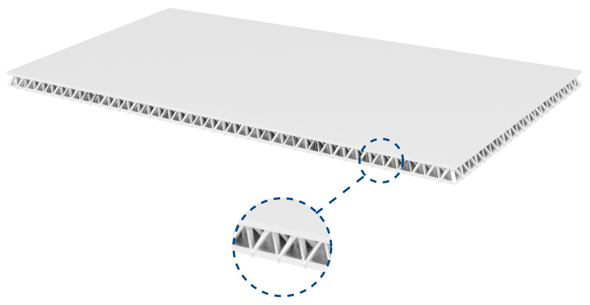
$$[\mathbf{G}^{\mathsf{T}}_{\text{c}}]_{ij} = N^{\text{s}}_i(\boldsymbol{\theta}^l_j)\mathbf{I}_3 \tag{4.34}$$

with $i \in \text{D}^{\text{s}}_{\text{c}}$ and $j \in \mathcal{D}^{\text{l}}_{\text{c}}$ the indices of coupled thin-shell nodes and coupled lattice nodes, respectively.

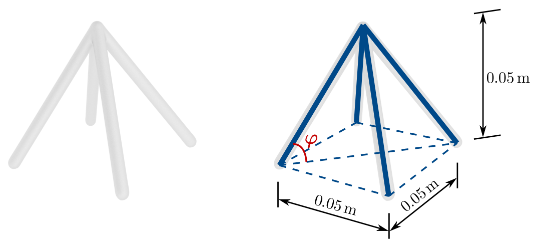
4.3 Examples

4.3.1 Sandwich plate

A simply-supported sandwich plate with pyramidal lattice core, as shown in Figure 4.2, is used to verify the accuracy and the convergence of the finite element computation



(a) Sandwich plate with pyramidal lattice core



(b) Unit cell with pyramidal tessellation

Fig. 4.2 Geometry of the sandwich plate.

proposed in the preceding section. The dimension of the sandwich plate is 1 m × 2 m, and the distance between the top and bottom thin-shell is 0.05 m. The pyramidal lattice core consists of cubic unit cells of side length 0.05 m and the struts are inclined by $\varphi = \arctan(\sqrt{2})$ with respect to the bottom skin, see Figure 4.2b. The both skins and the struts are made of the same material with a Young's modulus $E = 70\,\text{GPa}$ and a Poisson's ratio $\nu = 0.35$. The top skin is subjected to a uniform pressure loading of $7 \cdot 10^5\,\text{N/m}^2$. The bottom skin is at its edges simply supported and each skin is discretised with 256 Kirchhoff-Love shell elements and cubic b-splines.

The relative density $\bar{\rho}$ of a pyramidal lattice core is

$$\bar{\rho} = \frac{\pi}{2\cos^2\varphi\sin\varphi}\left(\frac{d}{L}\right)^2 , \tag{4.35}$$

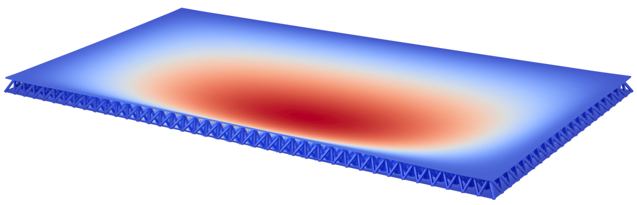
where d is the diameter of a lattice strut, L is the length of a strut.

The homogenised material properties for the pyramidal lattice core can be found in [102]. The shear modulus of a pyramidal core can be considered as isotropic is given by

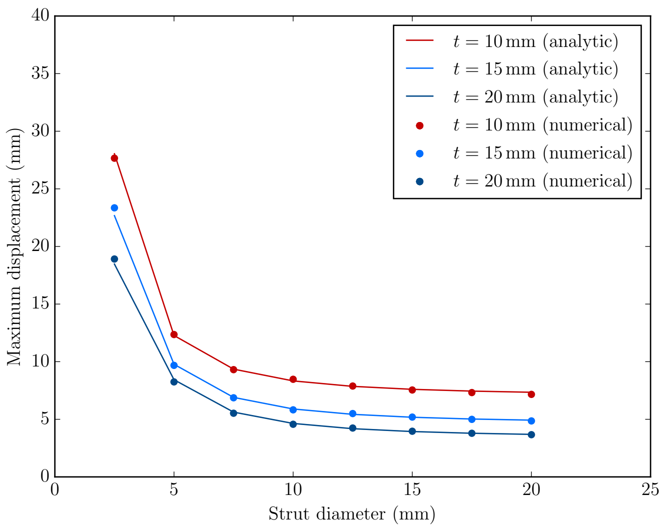
$$\bar{G} = \frac{\bar{\rho}}{8} E \sin^2 2\varphi \, . \tag{4.36}$$

Analytic expressions for the displacements of an isotropic sandwich plate under uniform loading can be found in [103]. The contribution of the lattice core to the flexural rigidity of the sandwich plate is neglected in analytic solutions.

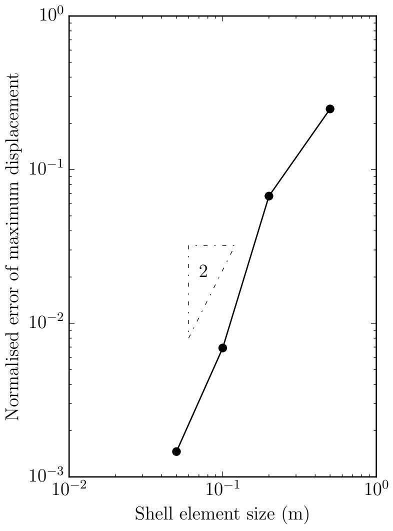
Three different skin thicknesses $t = \{10\,\text{mm}, 15\,\text{mm}, 20\,\text{mm}\}$ and eight different strut diameters $d = \{2.5\,\text{mm}, 5\,\text{mm}, 7.5\,\text{mm}, 10\,\text{mm}, 12.5\,\text{mm}, 15\,\text{mm}, 17.5\,\text{mm}, 20\,\text{mm}\}$ are considered. The comparison of the numerical and the analytic maximum displacements is plotted in Figure 4.3b. It can be seen that the numerical results agree well with the analytic ones for a range of different strut diameters and skin thicknesses. As depicted in Figure 4.3c the maximum displacement exhibits a quadratic convergence



(a) Deformation of the sandwich plate



(b) Comparison with analytic results



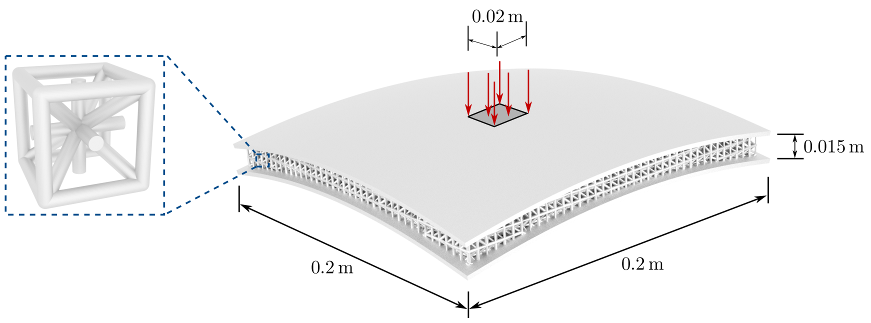
(c) Convergence of the maximum displacement

Fig. 4.3 Maximum displacement of the sandwich plate under uniform pressure loading.

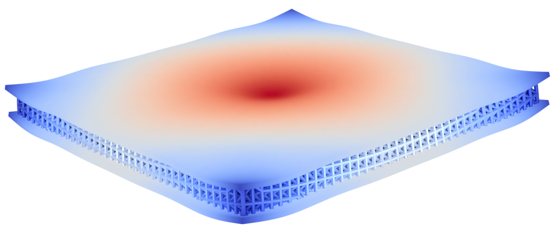
rate (for a plate with strut diameter $d = 17.5$ mm and skin thickness $t = 15$ mm). The convergence rates for different strut diameters and skin thicknesses may fluctuate around the quadratic rate since in the reference solution [103] the lattice core is approximated with an isotropic bulk material, and the skin-lattice coupling has also an effect on the convergence rate. A limitation of using truss elements to model the lattice in this example is that the shear and the moment are neglected in struts with high ratios of diameter to length greater than a threshold (for example $1/10 \sim 1/8$), but this can lead to slight difference with the real model as the lattice is stretch dominated.

4.3.2 Doubly curved sandwich panel

A curved sandwich panel with a BCC lattice core is shown in Figure 4.4a and the lattice edges are aligned with the principal axes of the cap. Each unit cell has a side length of 0.005 m and all struts have a diameter depending on the prescribed ratio of



(a) Geometry of the doubly curved sandwich panel



(b) Deformation under the loading

Fig. 4.4 Doubly-curved sandwich panel under the loading.

the lattice volume to the total material volume. The size of the sandwich cap projected onto the horizontal plane is 0.2 m $\times$ 0.2 m. The distance between the top and bottom skin is 0.015 m. Each skin is discretised with 64 Kirchhoff-Love shell elements and cubic b-splines. The Young's modulus and the Poisson's ratio of the thin-shell and the lattice material are $E = 100\,\text{GPa}$ and $\nu = 0$. A uniform loading with the magnitude $10^6\,\text{N/m}^2$ is applied in the region $[-0.01\,\text{m}, 0.01\,\text{m}] \times [-0.01\,\text{m}, 0.01\,\text{m}]$ around the centre of the upper skin. The four corners of the top and bottom skins are fixed. The deformation under the loading is shown in Figure 4.4b.

The total volume of the sandwich panel is prescribed and the optimal lattice to total volume ratio is sought. In Figure 4.5 the structural compliance for different lattice to total volume ratios from 0.07 to 0.85 are plotted. Three different total volumes $V \in \{150\,\text{cm}^3, 200\,\text{cm}^3, 250\,\text{cm}^3\}$ are considered, and the ranges of their corresponding strut diameters are 0.024 cm $\sim$ 0.084 cm, 0.028 cm $\sim$ 0.096 cm and 0.03 cm $\sim$ 0.108 cm, respectively. For each total volume there exists an optimal lattice to total volume ratio as can be inferred from Figure 4.5. Structures consisting either only of a lattice or only one shell skin are non-optimal. If the lattice constitutes a relatively large volume ratio in the sandwich panel, the compliance becomes larger as the lattice volume increases and even greater than the corresponding thin-shell structure with the

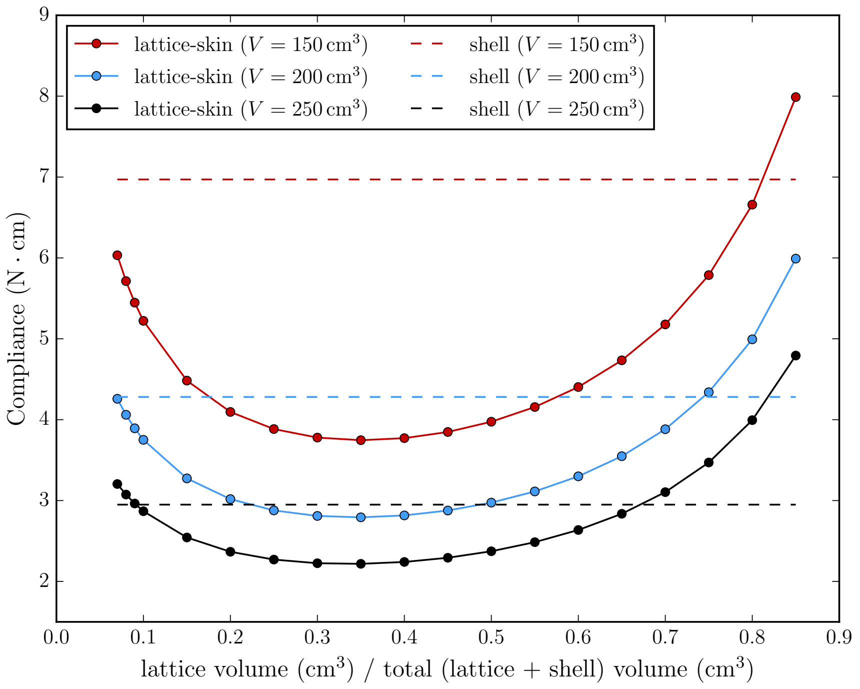


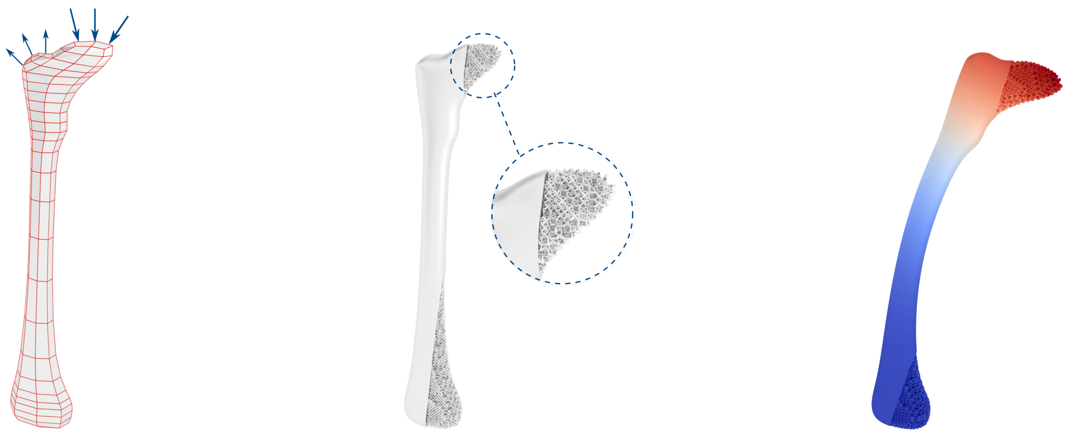
Fig. 4.5 Compliances with different ratios of lattice volume to total material volume.

same total material volume. It implies that the lattice structure may not be optimal compared with the solid structure with respect to the compliance, which has also been discussed recently in the topology optimisation literature [104]. When the lattice volume is relatively small, the lattice core becomes soft, which weakens the coupling effect of the top and the bottom thin-shells and impair the capability of carrying the shear force, leading to an increase in the compliance as shown in the results. It should be noted that modelling with truss elements may result in a stiffer response of the lattice structure having struts with diameter/length ratios greater than a threshold (for example, $1/8 \sim 1/10$, i.e. $0.05\,\mathrm{cm} \sim 0.0625\,\mathrm{cm}$ in this example), but the conclusion of the existence of an optimal ratio of lattice volume to total material volume would not be affected by the approximation of using truss elements.

4.3.3 Lattice-infilled femur bone

A femur bone surface parameterised with a Catmull-Clark subdivision surface is shown in Figure 4.6a. A BCC lattice structure is generated to infill the enclosed bone surface with the approach described in Section 3.3 as shown in Figure 4.6b. Each lattice unit cell has a side length of $0.005\,\mathrm{m}$. The femur bone is supposed to carry a loading of

$5 \times 10^4\,\mathrm{N/m^2}$. The Young's modulus of the material is 14 GPa. The surface of the femur bone is discretised with 232 Kirchhoff-Love shell elements and cubic B-splines.



(a) Control mesh (b) Lattice-infilled femur (c) Deformation

Fig. 4.6 Lattice infilled femur bone and deformation under the loading.

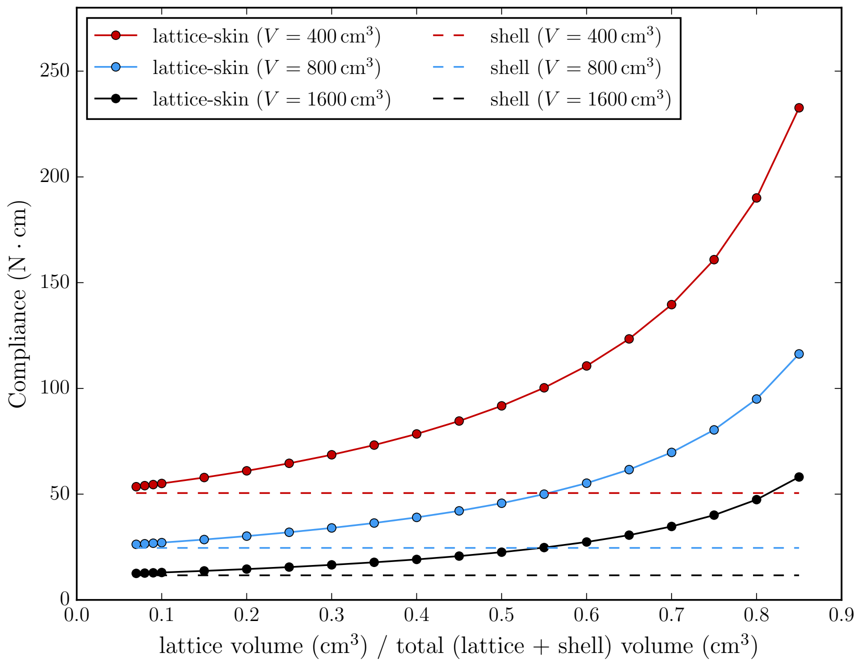


Fig. 4.7 Compliances with different ratios of lattice volume to total material volume.

Figure 4.7 shows the compliance of the lattice-infilled femur bone with different ratios of lattice volume to total material volume from 0.07 to 0.85. Three different total volumes $V \in \{400\,\mathrm{cm}^3, 800\,\mathrm{cm}^3, 1600\,\mathrm{cm}^3\}$ are considered, and the ranges of their corresponding strut diameters are $0.026\,\mathrm{cm} \sim 0.092\,\mathrm{cm}$, $0.037\,\mathrm{cm} \sim 0.13\,\mathrm{cm}$ and $0.052\,\mathrm{cm} \sim 0.184\,\mathrm{cm}$, respectively. Different from the results of the sandwich panel in the previous example, the compliance of the lattice-infilled femur bone is reduced as the lattice volume ratio increases, indicating that the lattice structure is not optimal in contrast to the solid with respect to the compliance. Since for the femur bone under the loading depicted as in Figure 4.6a, the stresses of the struts near the bone axis are very small, which means that there exists redundancy in struts which could have been removed in the structure, and this can explain the non-optimality of using infilled lattice structure for the femur bone model. For struts with ratios of diameter to length greater than a threshold (for example, $1/8 \sim 1/10$, i.e. $0.05\,\mathrm{cm} \sim 0.0625\,\mathrm{cm}$ in this example), the conclusion of the non-optimality of using lattice would not be affected though the approximation of using truss elements may result in a stiffer structural response in these cases.

Chapter 5

Optimisation of lattice structures

This chapter discusses the optimisation of lattice structures using gradient-based optimisation algorithms. Compared with gradient-free optimisation methods, for example genetic algorithms [105] and evolution strategies [106], the computational cost of the gradient-based algorithms is much lower, especially for large-scale optimisation problems with a large number of design variables. The gradient-based optimisation algorithms are briefly described in Appendix C, including the sequential quadratic programming (SQP) and the method of moving asymptotes (MMA). The size and shape optimisation of lattice structures are discussed in detail in Section 5.2 and the sensitivity analysis of the optimisation problem is derived.

In Section 5.3 a new lattice topology optimisation method based on lattice unit cells is proposed. The SIMP method in topology optimisation of continuum structures is adapted to be applicable in lattice topology optimisation, and several examples are given afterwards to demonstrate the feasibility of the method.

5.1 Gradient-based optimisation formulations

5.1.1 General description of structural optimisation

The general form of a structural optimisation problem can be described as

minimise

$$J(\boldsymbol{x}) \tag{5.1a}$$

subject to

$$G_i(\boldsymbol{x}) = 0, \quad \text{for } i = 1, \cdots, n_p \tag{5.1b}$$

$$H_j(\boldsymbol{x}) \leq 0, \quad \text{for } j = 1, \ \cdots, n_q \tag{5.1c}$$

where $J(\boldsymbol{x})$ is the objective function, which can be structural compliance, material volume, etc.; $G_i(\boldsymbol{x})$ are equality constraints, including structural equilibrium equations; $H_j(\boldsymbol{x})$ are inequality constraints, including the lower bound and the upper bound of design variables $\boldsymbol{x}$; n_p and n_q are the number of inequality and equality constraints, respectively.

The constraint functions can be considered with the Lagrangian of the optimisation problem as follows

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = J(\boldsymbol{x}) + \sum_{i=1}^{n_p} \lambda_i G_i(\boldsymbol{x}) + \sum_{j=1}^{n_q} \phi_j H_j(\boldsymbol{x}) \,, \tag{5.2}$$

where λ_i and ϕ_j are Lagrange multipliers, and ϕ_j must be nonnegative. By virtue of Lagrange multipliers, the original constrained optimisation problem is naturally converted to an unconstrained optimisation problem. At a local minimum of the constrained optimisation problem, it can be shown that the gradient of the objective function is a linear combination of gradients of constraint functions [107]. As a matter of fact, the coefficients of the linear combination are Lagrange multipliers in (5.2) which correspond to active constraints $G_i = 0$ and $H_j = 0$.

The necessary conditions for a feasible point $\boldsymbol{x}$ to be a local minimum of the constrained optimisation problem are summarised as the following Karush-Kuhn-Tucker (KKT) conditions [107],

$$\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = 0 \tag{5.3a}$$

$$G_i(\boldsymbol{x}) = 0, \quad H_j(\boldsymbol{x}) \leq 0 \tag{5.3b}$$

$$\lambda_i G_i(\boldsymbol{x}) = 0, \quad \phi_j H_j(\boldsymbol{x}) = 0 \tag{5.3c}$$

$$\phi_j \geq 0 \tag{5.3d}$$

where $i = 1, \cdots, n_p$ and $j = 1, \ \cdots, n_q$.

Conditions (5.3b) indicate that $\boldsymbol{x}$ must be a feasible point. Conditions (5.3a) and (5.3c) imply that the gradient of the objective function is a linear combination of active constraint functions, i.e.

$$\nabla_{\boldsymbol{x}} J(\boldsymbol{x}) + \sum_{i=1}^{n_p} \lambda_i \nabla_{\boldsymbol{x}} G_i(\boldsymbol{x}) + \sum_{j \in \mathcal{A}} \phi_j \nabla_{\boldsymbol{x}} H_j(\boldsymbol{x}) = 0 \,, \tag{5.4}$$

where $\mathcal{A}$ is the set of indices for which the constraints $H_j \leq 0$ are active, i.e. $H_j = 0$. It can also be seen from (5.4) that Lagrange multipliers reflect how sensitive the optimal value of the objective function is with respect to variations of constraints [108]. Specifically, for inequality constraints $H_j \leq 0$ which are inactive (i.e. $H_j < 0$), the corresponding Lagrange multipliers are zero, which implies that these constraints are not significant and small perturbations of these inactive constraints may not influence the optimal value of the objective function.

5.1.2 Sensitivity analysis

When solving the structural optimisation problem (5.1) with gradient-based optimisation algorithms, the first derivatives of objective and constraint functions with respect to design variables need to be computed. This process is usually referred to as *sensitivity analysis* in optimisation literature. For the objectives and constraints that are commonly considered in structural optimisation problems, the expressions of their derivatives will be given in detail in the following sections.

In structural optimisation problems, the general expression of the first derivatives with respect to design variables x_k can be written as

$$\frac{\partial f(\boldsymbol{x})}{\partial x_k} = \frac{\partial f(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial x_k} + \frac{\partial f(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_k}, \tag{5.5}$$

where $f(\boldsymbol{x})$ can be either the objective function $J(\boldsymbol{x})$ or constraint functions $G_i(\boldsymbol{x})$ or $H_i(\boldsymbol{x})$.

Considering the direct differentiation of the discretised structural equilibrium equations

$$\mathbf{K}(\boldsymbol{x})\mathbf{u}(\boldsymbol{x}) = \mathbf{F}(\boldsymbol{x}), \tag{5.6}$$

we have

$$\frac{\partial \mathbf{u}(\boldsymbol{x})}{\partial x_k} = \mathbf{K}^{-1}(\boldsymbol{x}) \left(\frac{\partial \mathbf{F}(\boldsymbol{x})}{\partial x_k} - \frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k} \mathbf{u}(\boldsymbol{x}) \right), \tag{5.7}$$

where $\mathbf{F}$ is the external force vector; $\mathbf{u}$ is the displacement vector; $\mathbf{K}$ is the global stiffness matrix of the structure, and the discretised global stiffness matrix $\mathbf{K}(\boldsymbol{x})$ can be expressed as

$$\mathbf{K}(\boldsymbol{x}) = \sum_{j=1}^{n_e} \mathbf{K}_j(\boldsymbol{x}), \tag{5.8}$$

$$\mathbf{K}_j(\boldsymbol{x}) = \mathbf{\Theta}_j^{\mathsf{T}} \mathbf{k}_j(\boldsymbol{x}) \mathbf{\Theta}_j, \tag{5.9}$$

where n_e is the number of finite elements; $\mathbf{K}_j$ is the global version of the element stiffness matrix $\mathbf{k}_j$, with the positions of nonzero entries in $\mathbf{K}_j$ corresponding to the degrees-of-freedom in the discretised displacement vector $\mathbf{u}$; $\mathbf{\Theta}_j$ is a scatter matrix containing entries 0 and 1, which maps the global discretised displacement vector $\mathbf{u}$ to the element displacement vector $\mathbf{u}_j$ with

$$\mathbf{u}_j = \mathbf{\Theta}_j \mathbf{u} \,. \tag{5.10}$$

The structural compliance is considered as the objective in this dissertation, i.e. $J(\boldsymbol{x}) = \mathbf{F}^\mathsf{T}\mathbf{u}$, and for the sake of simplicity, the external forces $\mathbf{F}$ are considered as independent of design variables, the first derivatives $\partial J(\boldsymbol{x})/\partial x_k$ in (5.5) become

$$\frac{\partial J(\boldsymbol{x})}{\partial x_k} = \mathbf{F}^\mathsf{T} \frac{\partial \mathbf{u}(\boldsymbol{x})}{\partial x_k} \,. \tag{5.11}$$

Substituting (5.7), (5.6), (5.8), (5.9) and (5.10) into (5.11) yields

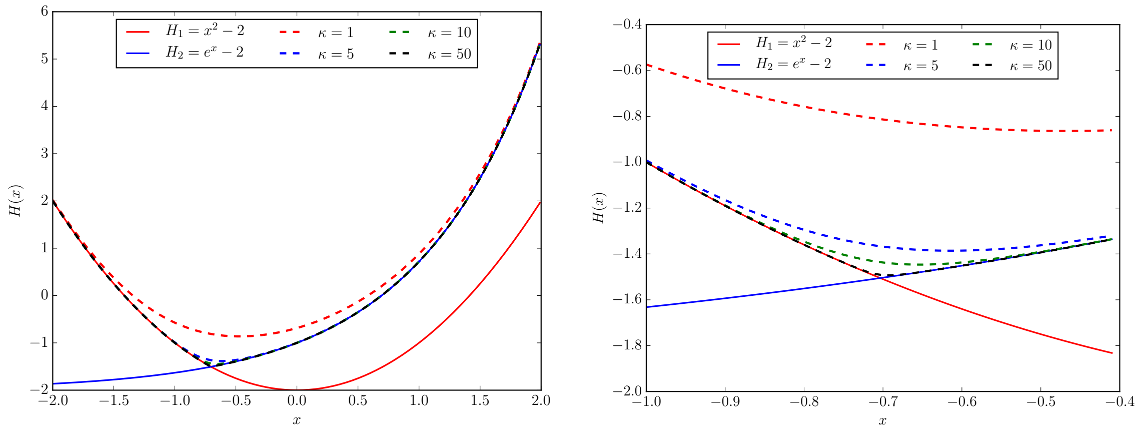
$$\frac{\partial J(\boldsymbol{x})}{\partial x_k} = -\mathbf{u}^\mathsf{T}(\boldsymbol{x}) \frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k} \mathbf{u}(\boldsymbol{x}) = -\sum_{j=1}^{n_e} \mathbf{u}_j^\mathsf{T}(\boldsymbol{x}) \frac{\partial \mathbf{k}_j(\boldsymbol{x})}{\partial x_k} \mathbf{u}_j(\boldsymbol{x}) \,. \tag{5.12}$$

The expression (5.12) may vary in details for different optimisation problems, for example size optimisation, shape optimisation and topology optimisation. The different expressions are derived in detail in the respective sections followed.

5.1.3 Constraint aggregation

For many practical optimisation problems, a large number of constraints need to be considered in general, for example, the lengths of lattice struts, the positions of lattice nodes, the stresses of lattice struts, etc.. As a result, the computational cost will be increased and the convergence of the optimisation would be affected when considering a large number of constraints for gradient-based optimisation algorithms. An ad hoc way to deal with a large number of constraints is to consider the constraint which is the most violated. This method is simple but has in general a poor convergence behaviour [109], as only one constraint is considered to determine the search direction in each iteration, which in general results in the violation of another constraint in the next iteration and leads to discontinuities in the optimisation problem. Therefore, it is appealing to aggregate multiple constraints into a single constraint function, especially when the constraint functions share common features.

One approach for the constraint aggregation is to use the Kreisselmeier-Steinhauser (KS) function as first proposed in [110]. The KS function used in our implementation



(a) KS functions with different parameters κ (b) Approximation of KS functions near the intersection region

Fig. 5.1 The KS functions for two constraints considering different parameters κ.

is given by

$$H_{\mathrm{ks}} = H_{\max}(\boldsymbol{x}) + \frac{1}{\kappa}\ln\left[\sum_{j}^{n_q} \mathrm{e}^{\kappa(H_j(\boldsymbol{x}) - H_{\max}(\boldsymbol{x}))}\right], \tag{5.13}$$

where $H_{\max}(\boldsymbol{x})$ is the most violated constraint, and κ is an aggregation parameter which indicates the extent of constraint aggregation to the most violated constraint. As the parameter κ increases, the KS function of the constraints approaches to $H_{\max}(\boldsymbol{x})$. Furthermore, the value of the KS function is bounded with

$$H_{\max}(\boldsymbol{x}) < H_{\mathrm{ks}} < H_{\max}(\boldsymbol{x}) + \frac{\ln n_q}{\kappa}, \tag{5.14}$$

which implies that the KS function is a conservative approximation to the constraint. Figure 5.1 gives an example to illustrate the approximation of constraint functions using the KS function.

The two constraint functions considered in Figure 5.1 are $H_1(x) = x^2 - 2$ and $H_2(x) = e^x - 2$, which are plotted in the domain $[-2, 2]$ as shown in Figure 5.1a. The KS functions with different parameters $\kappa = \{1, 5, 10, 50\}$ are also plotted. As can be seen, as the parameter κ increases, the KS function approaches to the constraint function which has the maximum value, i.e. the one most violated. The details of these different KS functions near the intersection region of the two constraints are shown in Figure 5.1b. In general, $\kappa = 50$ is a reasonable value which ensures the approximation of constraint functions as well as the numerical stability of the optimisation process. An adaptive KS function was also proposed in [109] which improves the accuracy of the approximation by selecting the parameter κ adaptively during the optimisation iterations to deal with the problem of overestimate using the conventional KS function (5.13).

5.2 Lattice size and shape optimisation

For the size and shape optimisation of lattice structures, we consider the structural compliance as the objective and the lattice volume as the constraint. The external force is assumed to be independent of design variables. The size and shape optimisation problem is described as

minimise

$$J(\boldsymbol{x}) = \mathbf{F}^{\mathsf{T}}\mathbf{u} \tag{5.15}$$

subject to

$$V(\boldsymbol{x}) = \sum_{i=1}^{n} A_i L_i \leq V_0 \tag{5.16}$$

$$\mathbf{K}\mathbf{u} = \mathbf{F} \tag{5.17}$$

where $\boldsymbol{x}$ are design variables, which are cross-sectional areas of struts in size optimisation and nodal coordinates in shape optimisation; A_i and L_i are the cross-sectional area and the length of the i-th strut respectively, V_0 is a prescribed lattice volume.

Lattice structures can be analysed with either truss elements or beam elements. For structures containing a large number of lattice struts in which the bending strain energy constitutes a very small fraction of the total strain energy, truss elements are adopted as the computational cost is much smaller. As a matter of fact, it is sufficient and valid to model the lattice structure with pin-jointed struts when the lattice structure is stretch-dominated [111, 99, 100] as is the case in this dissertation.

5.2.1 Size optimisation

In the context of pin-jointed trusses the compliance is a convex function, which is an appealing property as the size optimisation becomes a convex problem. That the compliance is a convex function is demonstrated as follows.

Recall (5.12) that the gradient of compliance reads

$$\frac{\partial J(\boldsymbol{x})}{\partial x_k} = -\mathbf{u}^{\mathsf{T}}(\boldsymbol{x})\frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k}\mathbf{u}(\boldsymbol{x})\,, \tag{5.18}$$

and the global stiffness matrix $\mathbf{K}$ of lattice structures with truss elements depends on cross-sectional areas linearly. The second derivatives of the compliance are

$$\begin{aligned}\frac{\partial^2 J(\boldsymbol{x})}{\partial x_k \partial x_l} &= -\left(\frac{\partial \mathbf{u}(\boldsymbol{x})}{\partial x_l}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k}\mathbf{u}(\boldsymbol{x}) - \mathbf{u}^{\mathsf{T}}(\boldsymbol{x})\frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k}\frac{\partial \mathbf{u}(\boldsymbol{x})}{\partial x_l} \\ &= 2\mathbf{u}^{\mathsf{T}}(\boldsymbol{x})\frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k}\mathbf{K}^{-1}(\boldsymbol{x})\frac{\mathbf{K}(\boldsymbol{x})}{\partial x_l}\mathbf{u}(\boldsymbol{x})\,. \end{aligned} \tag{5.19}$$

Hence, for any vector $\boldsymbol{y}$, we have

$$\begin{aligned}\boldsymbol{y}^{\mathsf{T}}\nabla^2_{\boldsymbol{x}} J(\boldsymbol{x})\boldsymbol{y} &= \sum_l \sum_k y_k \frac{\partial^2 J(\boldsymbol{x})}{\partial x_k \partial x_l} y_l \\ &= 2\mathbf{u}^{\mathsf{T}}(\boldsymbol{x})\left[\sum_l \sum_k \frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k} y_k \mathbf{K}^{-1}(\boldsymbol{x}) \frac{\mathbf{K}(\boldsymbol{x})}{\partial x_l} y_l\right]\mathbf{u}(\boldsymbol{x}) \\ &= 2\mathbf{u}^{\mathsf{T}}\left[\boldsymbol{Y}\mathbf{K}^{-1}(\boldsymbol{x})\boldsymbol{Y}\right]\mathbf{u}(\boldsymbol{x}) \\ &= 2\left(\boldsymbol{Y}\mathbf{u}(\boldsymbol{x})\right)^{\mathsf{T}}\mathbf{K}^{-1}(\boldsymbol{x})\left(\boldsymbol{Y}\mathbf{u}(\boldsymbol{x})\right) \geq 0\,, \end{aligned} \tag{5.20}$$

where the symmetric matrix $\boldsymbol{Y} = \sum_k \frac{\partial \mathbf{K}(\boldsymbol{x})}{\partial x_k} y_k$ is introduced. As can be seen, the Hessian of the compliance $\nabla_{\boldsymbol{x}} J(\boldsymbol{x})$ is always positive semi-definite. Therefore, the compliance is a convex function.

Sensitivity analysis

The element stiffness matrix of the j-th strut is

$$\mathbf{k}_j = \boldsymbol{\Lambda}_j^{\mathsf{T}}\overline{\mathbf{k}}_j\boldsymbol{\Lambda}_j\,, \tag{5.21}$$

where $\overline{\mathbf{k}}_j$ is the element stiffness matrix in the local coordinate system which only depends on the properties of the strut and it is linear with respect to the cross-sectional area when the truss element is used, $\boldsymbol{\Lambda}_j$ is a transformation matrix which maps the local stiffness matrix $\overline{\mathbf{k}}_j$ to $\mathbf{k}_j$ in the global coordinate system. For instance, the element stiffness matrix of a 2D truss element is

$$\mathbf{k}_j = \frac{E_j A_j}{L_j}\begin{bmatrix} c^2 & sc & -c^2 & -sc \\ sc & s^2 & -sc & -s^2 \\ -c^2 & -sc & c^2 & sc \\ -sc & -s^2 & sc & s^2 \end{bmatrix}\,, \tag{5.22}$$

where E_j, A_j and L_j are the Young's modulus, the cross-sectional area and the length of the j-th strut; $s = \sin\varphi$ and $c = \cos\varphi$ with φ indicating the orientation of the strut in the global coordinate system. When beam elements are used, the transformation matrix $\boldsymbol{\Lambda}_j$ is given in Appendix D.

The sensitivity of the compliance (5.12) becomes

$$\frac{\partial J(\boldsymbol{A})}{\partial A_j} = -\mathbf{u}_j^{\mathsf{T}} \frac{\partial \mathbf{k}_j}{\partial A_j} \mathbf{u}_j \,, \tag{5.23}$$

where $\mathbf{u}_j$ is the displacement vector of the j-th strut, and the first derivative of the element stiffness matrix with respect to the cross-sectional area is straightforward to compute with

$$\frac{\partial \mathbf{k}_j}{\partial A_j} = \boldsymbol{\Lambda}_j^{\mathsf{T}} \frac{\partial \overline{\mathbf{k}}_j}{\partial A_j} \boldsymbol{\Lambda}_j \,. \tag{5.24}$$

In addition, the sensitivity of the volume constraint is readily obtained as

$$\frac{\partial (V(\boldsymbol{A}) - V_0)}{\partial A_j} = L_j \,. \tag{5.25}$$

Examples

A four-strut statically determinate truss A statically determinate truss structure with four struts as sketched in Figure 5.2 is considered to verify the size optimisation algorithm by comparing with the analytical optimisation solution. The structural compliance of the truss is given by

$$J(A_1, A_2, A_3, A_4) = \frac{Fs}{E} \left(\frac{1}{A_3} + \frac{2\sqrt{2}}{A_4} \right) \,, \tag{5.26}$$

where $A_i (i = 1, 2, 3, 4)$ are cross-sectional areas of the four struts with indices given in Figure 5.2, and E is the Young's modulus of the material.

The volume constraint is considered in the size optimisation,

$$V = (A_1 + A_2 + A_3 + \sqrt{2} A_4) \cdot s \leq V_0 \tag{5.27}$$

with V_0 the prescribed volume of the truss. In addition, the area A_i of each strut is required to be positive.

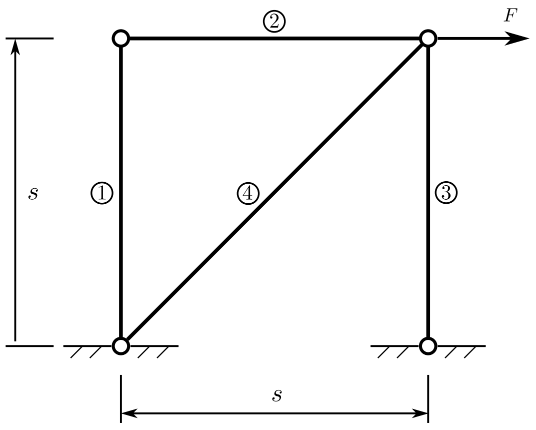


Fig. 5.2 A sketch of the four-strut statically determinate truss.

The Lagrangian of the size optimisation problem reads

$$\mathcal{L}(\boldsymbol{A},\boldsymbol{\mu}) = J(\boldsymbol{A}) + \phi_0(V - V_0) - \sum_{i=1}^{4} \phi_i A_i \tag{5.28}$$

with $\phi_i (i = 0, \cdots, 5)$ the Lagrange multipliers. The KKT conditions (5.3) of the size optimisation problem read

$$\phi_0 - \phi_1 = 0\,, \tag{5.29a}$$

$$\phi_0 - \phi_2 = 0\,, \tag{5.29b}$$

$$-\frac{Fs}{E}\frac{1}{A_3^2} + \phi_0 s - \phi_3 = 0\,, \tag{5.29c}$$

$$-\frac{Fs}{E}\frac{2\sqrt{2}}{A_4^2} + \sqrt{2}\phi_0 s - \phi_4 = 0\,, \tag{5.29d}$$

$$(A_1 + A_2 + A_3 + \sqrt{2}A_4) \cdot s - V_0 \leq 0\,, \tag{5.29e}$$

$$A_i \geq 0\,, \quad i = 1, 2, 3, 4,\ , \tag{5.29f}$$

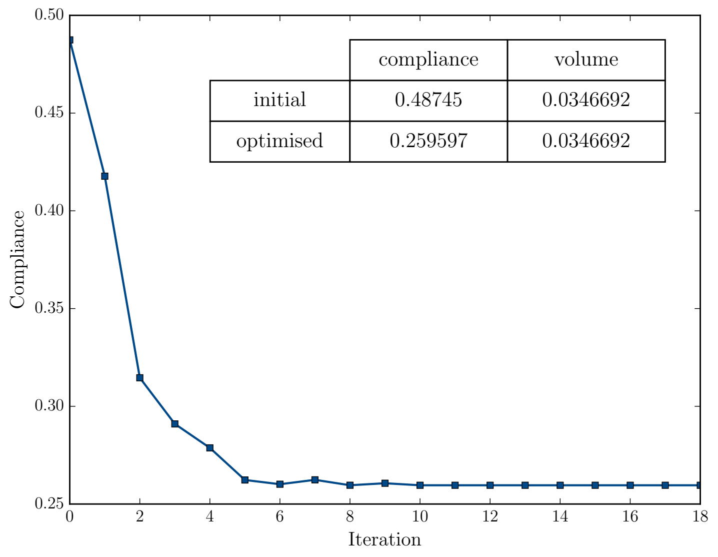
$$\phi_0 \left[(A_1 + A_2 + A_3 + \sqrt{2}A_4) \cdot s - V_0 \right] = 0\,, \tag{5.29g}$$

$$\phi_i A_i = 0\,, \quad i = 1, 2, 3, 4\,, \tag{5.29h}$$

$$\phi_i \geq 0\,, \quad i = 0, 1, 2, 3, 4\,. \tag{5.29i}$$

As a result, the equation system (5.29) has a solution only when $\phi_0, \phi_1, \phi_2 \neq 0$ and $\phi_3 = \phi_4 = 0$. Substituting $\phi_3 = \phi_4 = 0$ into the KKT conditions, the solution is readily obtained as

$$A_1 = 0\,, \quad A_2 = 0\,, \quad A_3 = \frac{V_0}{3s}\,, \quad A_4 = \frac{\sqrt{2}V_0}{3s} \tag{5.30}$$



	compliance	volume
initial	0.48745	0.0346692
optimised	0.259597	0.0346692

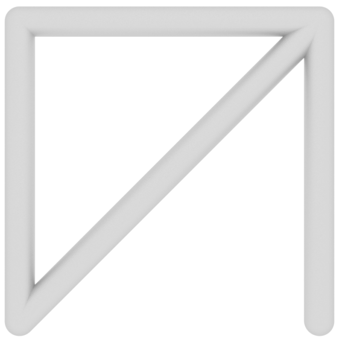
Fig. 5.3 Convergence of the size optimisation of the four-strut truss.

with the corresponding minimum compliance $J_{\min} = 9Fs^2/EV_0$.

Next, the size optimisation of the truss is performed using a gradient-based optimisation algorithm. The lower bound of cross-sectional areas of struts is set to be a very small value, for example 10^{-8}, in order to avoid a singular stiffness matrix if the cross-sectional area of a strut becomes 0. The struts of the initial truss are assumed to have the same cross-sectional area with the diameter 0.1. The Young's Modulus $E = 1000$ and the external force $F = 1$.

The convergence of the size optimisation process is plotted in Figure 5.3. The optimisation process terminates after 18 iterations with the criterion that the relative change of cross-sectional areas is less than 10^{-6} satisfied. The compliance of the size optimised truss is 0.259597, reduced by 46.7% compared with that of the initial truss which is 0.48745.

In the end, the struts with cross-sectional areas less than a small threshold, for example 10^{-8} used in this example, are removed from the optimisation result. The final size optimised truss is shown in Figure 5.4. The comparison between the optimisation result using the optimisation algorithm developed and the analytical solution (5.30) is given in Table 5.1. It can be seen that the numerical result agrees very well with the analytical solution, and the cross-sectional areas of strut 1 and strut 2 converge to the prescribed lower bound which are removed in the final optimised truss.


(a) Initial four-strut truss

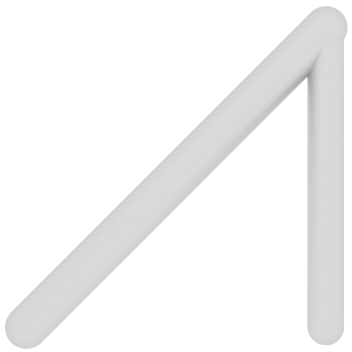
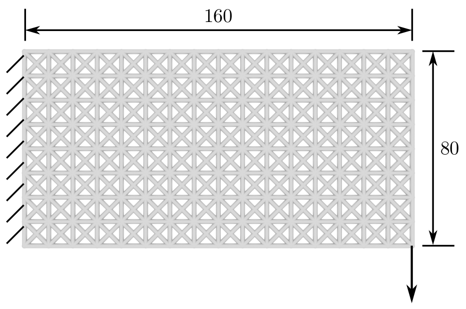

(b) Size optimised truss

Fig. 5.4 Size optimisation of the four-strut truss.

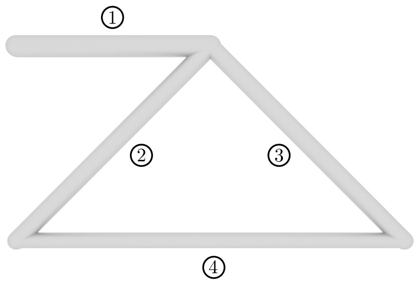
Table 5.1 Size optimisation results of the four-strut truss.

	compliance	A_1	A_2	A_3	A_4
initial	0.48745	0.0025π	0.0025π	0.0025π	0.0025π
analytic	0.259597	0	0	0.0115564	0.0163432
numerical	0.259597	10^{-8}	10^{-8}	0.0115564	0.0163432

A planar cantilever truss The planar truss considered for the size optimisation is depicted in Figure 5.5a. It is fixed on its left side and a unit external force is applied at the right bottom corner as is shown. The dimension of the planar truss is 160×80. The Young's modulus of the material is set as $E = 1000$, and the struts are assumed to have the same initial cross-sectional area with the diameter 2.



(a) Initial cantilever truss

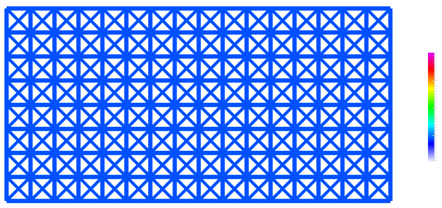


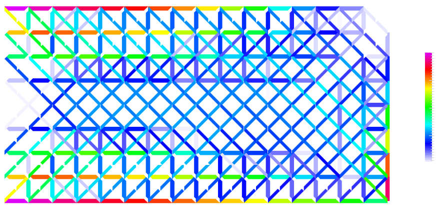
(b) Size optimised truss

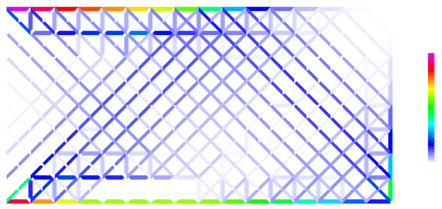
Fig. 5.5 Size optimisation result of the cantilever truss.

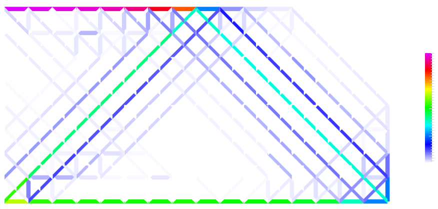
The volume constraint is that the lattice volume remains the same as the volume of the initial truss. Figure 5.6 gives the variation of cross-sectional areas of lattice struts during the optimisation iterations, with colours indicating the relative cross-sectional area values. The optimisation process converges after around 1600 iterations with the termination criterion that the cross-sectional areas of lattice struts vary less than 10^{-4}.

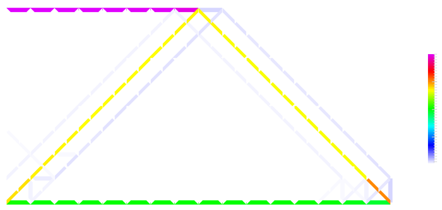
After removing lattice struts with very small cross-sectional areas (e.g. 10^{-3} used in this example), the optimised truss is shown in Figure 5.5b. The compliance of the optimised truss is 0.0203072, reduced by 71.2% compared with that of the initial cantilever truss which is 0.0703943. The optimal cross-sectional areas of the final truss are 63.045, 44.571, 44.571 and 31.516, respectively, and the corresponding diameters are 4.48, 3.767, 3.767 and 3.167, respectively.


(a) Initial cross-sectional areas


(b) 10 iterations


(c) 50 iterations


(d) 200 iterations


(e) 500 iterations

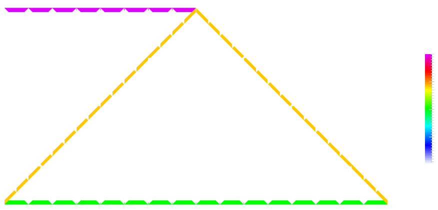
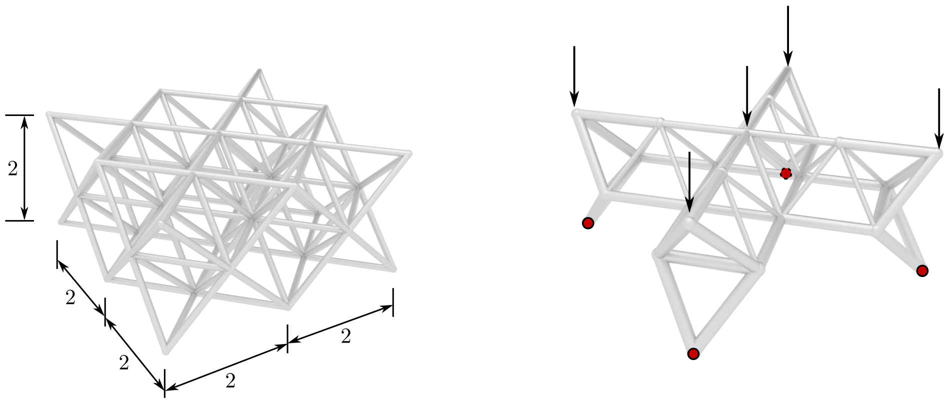

(f) Converged cross-sectional areas

Fig. 5.6 Cross-sectional areas of the cantilever truss at different iterations.

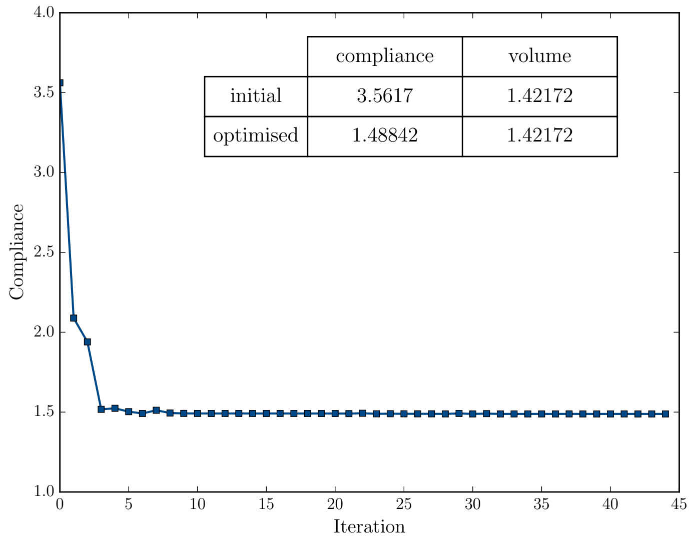
A spatial truss A spatial truss with octet truss cores as shown in Figure 5.7a is considered. The dimension of the initial spatial truss is $4 \times 2 \times 2$ comprised of four octet truss units. The truss is fixed at the four bottom corners and subjected to point loads with the magnitude of 1 at the nodes shown in Figure 5.7b. The Young's modulus of the material is set as $E = 1000$, and the initial cross-sectional areas of lattice struts are set to be equal with diameters 0.1.



(a) Initial spatial truss (b) Size optimised truss

Fig. 5.7 Size optimisation of the spatial truss.

The size optimised truss is shown in Figure 5.7b with the same lattice volume as the initial truss. The minimum and maximum diameters of the struts in the optimised truss are 0.0417 and 0.0932, respectively. The compliance of the optimised truss is 1.48842, reduced by 58.2% compared with that of the initial truss which is 3.5617. The convergence of the size optimisation process is plotted in Figure 5.8.



	compliance	volume
initial	3.5617	1.42172
optimised	1.48842	1.42172

Fig. 5.8 Covergence of the size optimisation of the spatial truss.

5.2.2 Shape optimisation

Sensitivity analysis

The gradient of the compliance function (5.12) in shape optimisation can be written as

$$\frac{\partial J(\boldsymbol{x})}{\partial x_p^q} = -\sum_j \mathbf{u}_j^\mathsf{T}(\boldsymbol{x}) \frac{\partial \mathbf{k}_j(\boldsymbol{x})}{\partial x_p^q} \mathbf{u}_j(\boldsymbol{x}) \,, \tag{5.31}$$

where p is the node index, q is the local index of the degree-of-freedom ($\{1, 2\}$ for 2D and $\{1, 2, 3\}$ for 3D), j denotes the struts which the node p belongs to, $\mathbf{u}_j$ and $\mathbf{k}_j$ are the displacement vector and the element stiffness matrix of the j-th strut respectively.

For truss structures, the element stiffness matrix can be written as

$$\mathbf{k}_j = \boldsymbol{\Phi}_j^\mathsf{T} D_j \boldsymbol{\Phi}_j \,, \tag{5.32}$$

where $D_j = E_j A_j / L_j$ contains the material properties of the strut which share the same symbols as in the size optimisation, $\boldsymbol{\Phi}_j$ is a matrix indicating the orientation of the strut, i.e.

$$\boldsymbol{\Phi}_j = \begin{pmatrix} -\boldsymbol{t}_j^\mathsf{T} & \boldsymbol{t}_j^\mathsf{T} \end{pmatrix} \tag{5.33}$$

with $\boldsymbol{t}_j$ a unit vector along the direction of the j-th strut, that is

$$\boldsymbol{t}_j = \frac{1}{L_j} \begin{pmatrix} x_{j,2}^1 - x_{j,1}^1 & x_{j,2}^2 - x_{j,1}^2 \end{pmatrix}^\mathsf{T} \quad \text{in 2D} \tag{5.34}$$

and

$$\boldsymbol{t}_j = \frac{1}{L_j} \begin{pmatrix} x_{j,2}^1 - x_{j,1}^1 & x_{j,2}^2 - x_{j,1}^2 & x_{j,2}^3 - x_{j,1}^3 \end{pmatrix}^\mathsf{T} \quad \text{in 3D.} \tag{5.35}$$

Hence, the first derivatives of the element stiffness matrix (5.32) with respect to nodal coordinates are computed with

$$\frac{\partial \mathbf{k}_j}{\partial x_p^q} = \frac{\partial \boldsymbol{\Phi}_j^\mathsf{T}}{\partial x_p^q} D_j \boldsymbol{\Phi}_j + \boldsymbol{\Phi}_j^\mathsf{T} \frac{\partial D_j}{\partial x_p^q} \boldsymbol{\Phi}_j + \boldsymbol{\Phi}_j^\mathsf{T} D_j \frac{\partial \boldsymbol{\Phi}_j}{\partial x_p^q} \tag{5.36}$$

with

$$\frac{\partial \boldsymbol{t}_j^\mathsf{T}}{\partial x_p^q} = \frac{1}{L_j} \boldsymbol{e}^\mathsf{T} (\mathbf{I} - \boldsymbol{t}_j \boldsymbol{t}_j^\mathsf{T}) \tag{5.37}$$

$$\frac{\partial \boldsymbol{\Phi}_j}{\partial x_p^q} = \begin{pmatrix} -\frac{1}{L_j} \boldsymbol{e}^\mathsf{T} (\mathbf{I} - \boldsymbol{t}_j \boldsymbol{t}_j^\mathsf{T}) & \frac{1}{L_j} \boldsymbol{e}^\mathsf{T} (\mathbf{I} - \boldsymbol{t}_j \boldsymbol{t}_j^\mathsf{T}) \end{pmatrix} \tag{5.38}$$

$$\frac{\partial D_j}{\partial x_p^q} = -\frac{E_j A_j}{L_j^2} \boldsymbol{e}^\mathsf{T} \boldsymbol{t}_j \,, \tag{5.39}$$

where

$$\boldsymbol{e} = \begin{cases} (-1 \quad 0)^\mathsf{T}, & \text{for } x_{j,1}^1 \\ (0 \quad -1)^\mathsf{T}, & \text{for } x_{j,1}^2 \\ (1 \quad 0)^\mathsf{T}, & \text{for } x_{j,2}^1 \\ (0 \quad 1)^\mathsf{T}, & \text{for } x_{j,2}^2 \end{cases} \quad \text{in 2D} \tag{5.40}$$

and

$$\boldsymbol{e} = \begin{cases} (-1 \quad 0 \quad 0)^\mathsf{T}, & \text{for } x_{j,1}^1 \\ (0 \quad -1 \quad 0)^\mathsf{T}, & \text{for } x_{j,1}^2 \\ (0 \quad 0 \quad -1)^\mathsf{T}, & \text{for } x_{j,1}^3 \\ (1 \quad 0 \quad 0)^\mathsf{T}, & \text{for } x_{j,2}^1 \\ (0 \quad 1 \quad 0)^\mathsf{T}, & \text{for } x_{j,2}^2 \\ (0 \quad 0 \quad 1)^\mathsf{T}, & \text{for } x_{j,2}^3 \,. \end{cases} \quad \text{in 3D}\,. \tag{5.41}$$

In addition, the sensitivity of the volume constraint is

$$\frac{\partial (V(\boldsymbol{x}) - V_0)}{\partial x_p^q} = \sum_j A_j \frac{\partial L_j(\boldsymbol{x})}{\partial x_p^q} = \sum_j A_j \boldsymbol{e}^\mathsf{T} \boldsymbol{t}_j \,. \tag{5.42}$$

An iterative size/shape optimisation scheme

For shape optimisation of the lattice structure modelled with truss elements as described in the preceding section, the lattice structure must not be a mechanism in the optimisation process, which unfortunately cannot be guaranteed when truss elements are used. However, the detection of mechanism in the structure is not trivial, especially for spatial lattice structures. In addition, it is not straightforward to cope with the possible existence of mechanism in the structure. Alternatively, beam elements can be used to model the lattice structure such that the requirement of being not a mechanism can be removed. When the lattice structure is modelled with beam elements, the lattice nodes can be merged if they are close enough in the space, which can be realised in a straightforward way by removing elements with very small lengths, so that the topology of the lattice structure is updated at the same time.

Furthermore, an iterative size and shape optimisation scheme can be performed to the lattice structure when beam elements are used. The lattice structure is first optimised for the beam element sizes with the nodal coordinates fixed, and the shape optimisation is then performed on the size optimised structure. The size/shape iterations continue until the optimised objective function value does not decrease. During the shape optimisation process, the lattice nodes are merged if the distances between them become zero, and the shape optimisation process continues with the new topology of the lattice structure. After each shape optimisation process, the beam elements with very small lengths are removed from the structure, i.e. coalescing the end nodes of those elements. The subsequent size optimisation is performed on the lattice structure with the updated topology.

The sensitivity analysis of the objective is the same as (5.23) and (5.31) with the beam element stiffness matrix used in these expressions. Recall the element stiffness matrix (5.21), the first derivatives of the element stiffness matrix with respect to the nodal coordinates are computed with

$$\frac{\partial \mathbf{k}_j}{\partial x_p^q} = \frac{\partial \mathbf{\Lambda}_j^\mathsf{T}}{\partial x_p^q}\bar{\mathbf{k}}_j\mathbf{\Lambda}_j + \mathbf{\Lambda}_j^\mathsf{T}\frac{\partial \bar{\mathbf{k}}_j}{\partial x_p^q}\mathbf{\Lambda}_j + \mathbf{\Lambda}_j^\mathsf{T}\bar{\mathbf{k}}_j\frac{\partial \mathbf{\Lambda}_j}{\partial x_p^q}\,, \tag{5.43}$$

where the first derivatives of the transformation matrix with respect to nodal coordinates $\partial\mathbf{\Lambda}_j/\partial x_p^q$ are derived in Appendix D. The beam element stiffness matrix is a combination of stiffness submatrices corresponding to stretch, bending and torsion respectively. Therefore, the first derivatives of the beam element stiffness are also a combination of the first derivatives of these three submatrices.

Examples

A two-strut truss The simplest truss with two struts as sketched in Figure 5.9 is considered to verify the shape optimisation algorithm by comparing with the analytical optimisation result. The aim of shape optimisation is to optimise the position of node 2 such that the compliance of the truss is minimised. The analytical solution of shape optimisation of the two-strut truss is derived as follows.

Assume the coordinates of the two support nodes, node 0 and node 1, be $(0,0)$ and $(2s,0)$, respectively, with s the half distance between the two support nodes. The unknown coordinate (x,y) of node 2 is to be optimised. A unit force is applied at node 2 as shown in Figure 5.9. It is assumed that node 2 can move freely in the space. The analytical expression of the compliance of the truss under the force is given by

$$J(x,y) = \frac{(2s-x)^2(x^2+y^2)\sqrt{x^2+y^2} + x^2\Big[(2s-x)^2+y^2\Big]\sqrt{(2s-x)^2+y^2}}{EA\Big[y(2s-x)+xy\Big]^2}\,, \tag{5.44}$$

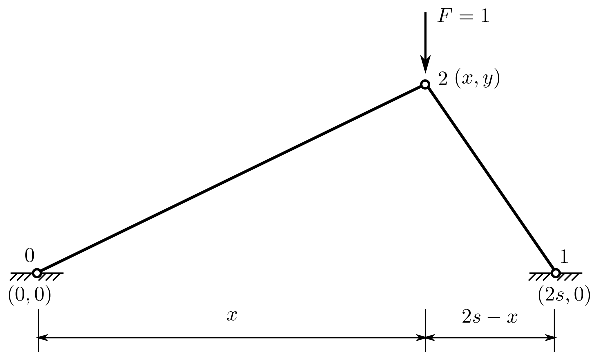
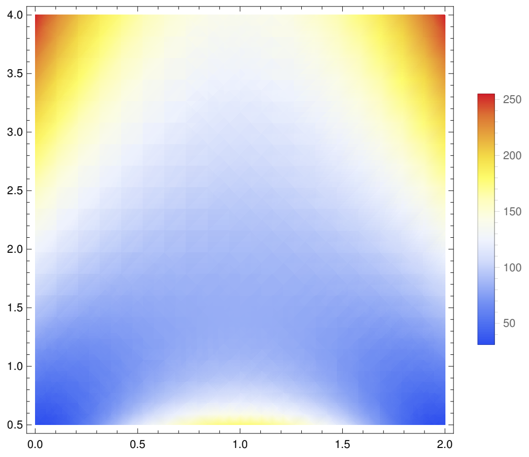


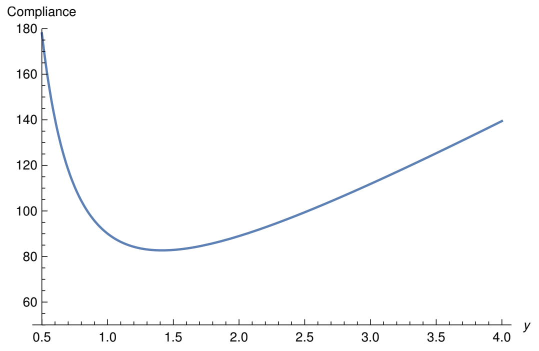
Fig. 5.9 A sketch of the two-strut truss.

where E and A are the Young's modulus and the cross-sectional area of truss elements.

Figure 5.10a shows the isocontours of the compliance for different positions of node 2 varying in the domain $\{(x, y)|x \in [0, 2], y \in [0.5, 4]\}$, with $s = 1$, $E = 2$ and diameters of the struts 0.1. The compliance along $x = 1$ is plotted in Figure 5.10b which indicates that there exists a position along $x = 1$ corresponding to a local minimum of the compliance.



(a) Compliance w.r.t. positions of node 2



(b) Compliance for node 2 moving along $x = 1$

Fig. 5.10 Analytical compliance of the truss with different positions of node 2.

If node 2 is restricted to move vertically along $x = s$, substituting $x = s$ into the compliance expression (5.44) yields

$$J(y) = \frac{1}{EA} \frac{(s^2 + y^2)\sqrt{s^2 + y^2}}{2y^2}, \tag{5.45}$$

which can be easily proved that $J(y)$ has a minimum at $y = \sqrt{2}s$ with the corresponding compliance $J_{\text{min}} = 3\sqrt{3}s/4EA$.

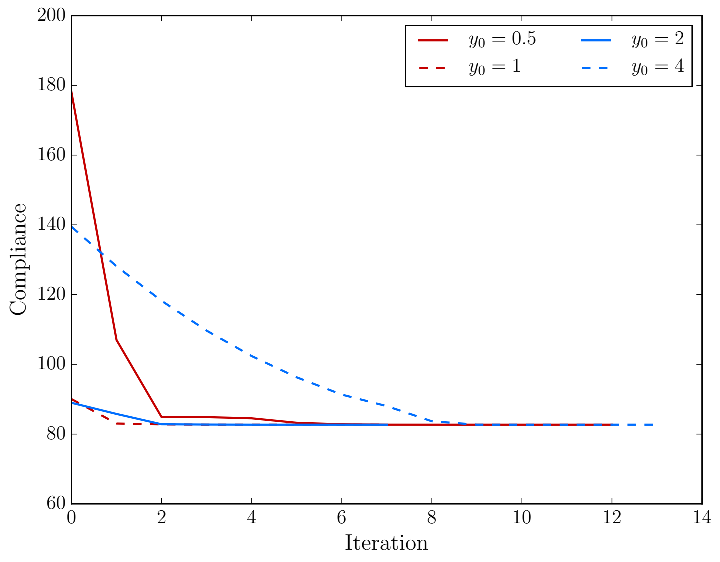


Fig. 5.11 Convergence of shape optimisation of the two-strut truss considering different initial positions.

Table 5.2 Shape optimisation results of the two-strut truss.

	compliance (Initial)	y_{opt}	compliance (Optimised)	#iterations
analytical	-	$\sqrt{2}$	82.6993	-
$y_0 = 0.5$	177.941	1.41421	82.6993	12
$y_0 = 1$	90.0316	1.41421	82.6993	7
$y_0 = 2$	88.9703	1.41421	82.6993	7
$y_0 = 4$	139.445	1.41421	82.6993	13

In order to verify the shape optimisation algorithm, the two-strut truss considered for shape optimisation takes $s = 1$. Node 2 is restricted to move vertically along $x = 1$. The analytical solution for the optimal position of node 2 is $y = \sqrt{2}$ and the corresponding compliance is 82.6993 according to (5.45). Different initial positions $(1, y_0)$ of node 2 are considered with $y_0 \in \{0.5, 1, 2, 4\}$. The optimised results are listed in Table 5.2. It can be seen from the optimisation results that the optimal position of node 2 obtained with the shape optimisation algorithm is coincident with the analytical solution, and the optimal position is the same considering different initial positions, indicating that it is an optimum in its neighbourhood, which is the fact also shown in Figure 5.10b. The convergence of the optimisation process considering different initial positions of node 2 is plotted in Figure 5.11. It can be observed that it takes fewer iterations if the initial position is closer to the optimal position.

A planar truss A planar truss sketched as in Figure 5.12 is considered for the shape optimisation. The truss is pinned at the two lower end nodes and subjected to nodal forces with the magnitude of 0.1 applied at nodes on the top. The Young's modulus is set as $E = 100$, and the cross-sectional areas of struts are assumed to be the same with diameters 0.1.

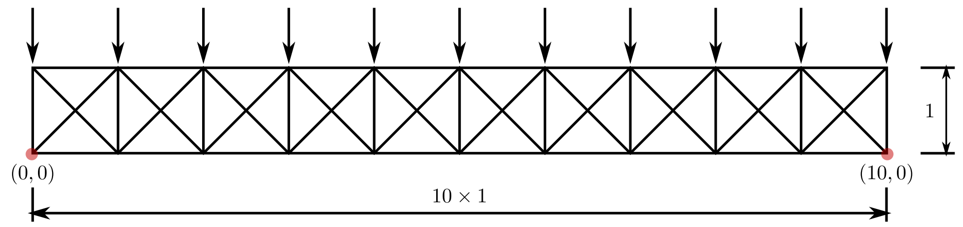
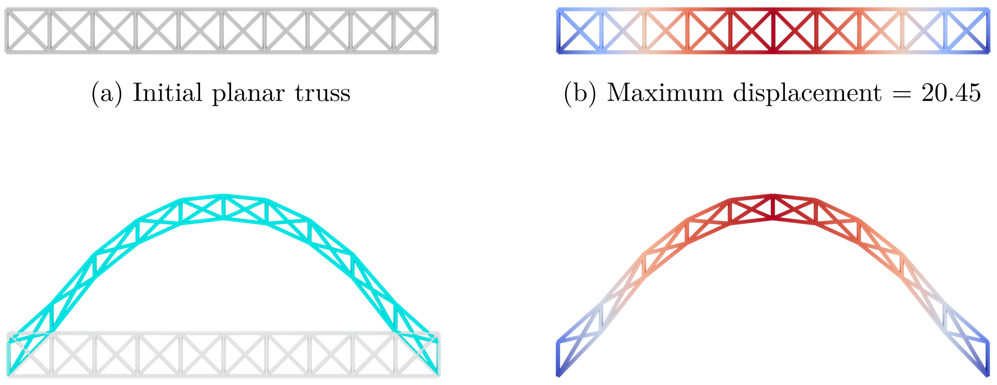


Fig. 5.12 A sketch of the planar truss.

The structural compliance of the initial planar truss as shown in Figure 5.13a is 12.6119 with the maximum displacement of 20.45. The lattice volume constraint is considered such that the lattice volume remains the same as the initial volume, that is, to minimise the compliance without increasing the material volume. In addition, the length L_j of each strut is restricted to be $0.5L_j^0 \leq L_j \leq 2L_j^0$ with L_j^0 the initial length of the j-th strut. The length constraints are applied with the constraint aggregation method using the KS approximation described in Section 5.1.3.

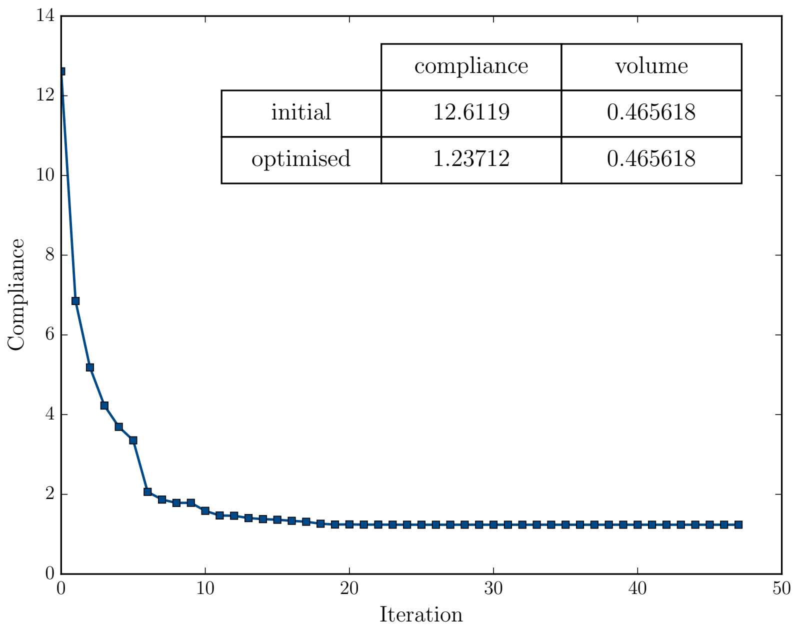


(a) Initial planar truss (b) Maximum displacement = 20.45

(c) Optimised planar truss (d) Maximum displacement = 1.79

Fig. 5.13 Shape optimisation result of the planar truss.

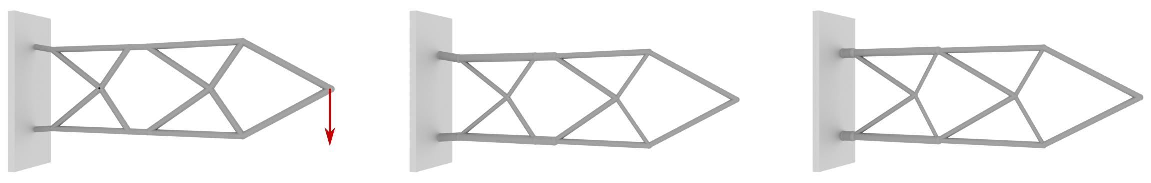
The shape optimised planar truss under these constraints is shown in Figure 5.13c with the maximum displacement of 1.79. The optimised compliance is 1.23712, which is reduced by 90.2% compared with that of the initial planar truss. The shape optimisation iterations are plotted in Figure 5.14. The shape optimisation terminates after 47 iterations with the termination criterion that the relative change of nodal coordinates is less than 10^{-6}.



	compliance	volume
initial	12.6119	0.465618
optimised	1.23712	0.465618

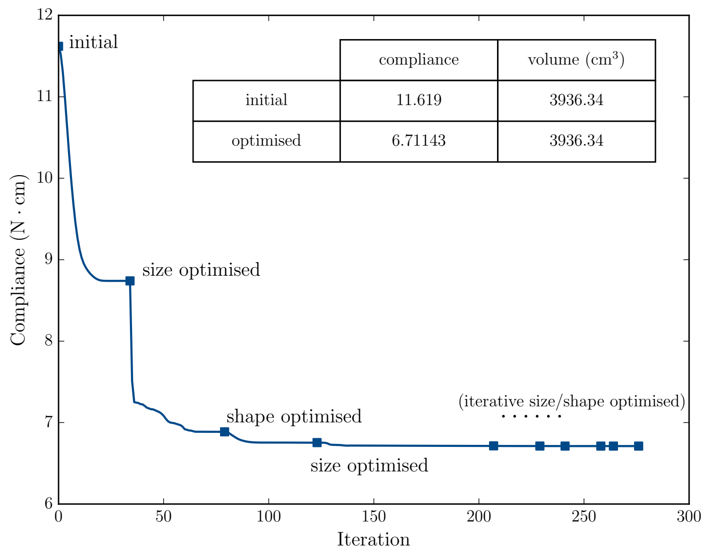
Fig. 5.14 Convergence of the shape optimisation process.

A cantilever lattice A cantilever lattice with the initial configuration as shown in Figure 5.15a is in the domain $[0\,\text{m}, 1.5\,\text{m}] \times [0\,\text{m}, 0.44\,\text{m}]$. The beam structure is adapted from the topology optimisation result of a continuum structure [112], which contains short beams and unsymmetrical features. A force with the magnitude of 1000 N is applied at the tip node on the right side. The cross-sectional areas of the beams in the initial cantilever are assumed to be the same, and initial diameters are 30 mm. The Young's modulus and the Poisson's ratio of the material are 210 GPa and 0.3, respectively.



(a) Initial cantilever beam (b) 1st size optimisation (c) Final optimised cantilever

Fig. 5.15 Iterative size/shape optimisation of a cantilever beam.



	compliance	volume (cm^3)
initial	11.619	3936.34
optimised	6.71143	3936.34

Fig. 5.16 Convergence of compliance in iterative size/shape optimisation process of the cantilever beam.

The size optimisation is first applied to the initial cantilever beam with the volume constraint that the size optimised cantilever has the same volume as the initial one. The size optimised cantilever is shown in Figure 5.15b with the compliance $8.74\,\mathrm{N}\cdot\mathrm{cm}$, reduced by 24.8% compared with that of the initial cantilever beam which is $11.619\,\mathrm{N}\cdot\mathrm{cm}$. The subsequent shape and size optimisation iterations are performed successively while keeping the same volume as the initial cantilever beam. In the shape optimisation process, the nodes are confined to move within the original domain of the continuum structure. The iterative optimisation process terminates when the optimised compliances of two consecutive size/shape optimisation processes have a difference less than a small tolerance, for example, 10^{-4} is used in this example.

The convergence of the iterative optimisation process is plotted in Figure 5.16. The entire iterative optimisation process involves five size optimisation iterations and four shape optimisation iterations. The compliance in the last optimisation process is reduced from $6.71146\,\mathrm{N}\cdot\mathrm{cm}$ to $6.71143\,\mathrm{N}\cdot\mathrm{cm}$, which has a difference less than 10^{-4}, leading to the termination of the iterative size/shape optimisation process. The final optimised cantilever is shown in Figure 5.15c with the compliance $6.711\,\mathrm{N}\cdot\mathrm{cm}$, reduced by 42.24% compared with that of the initial cantilever beam. As a result, the short beams in the original structure are removed automatically in the shape optimisation iterations, and the overall layout of struts becomes symmetric in the final optimised structure.

5.3 Lattice topology optimisation

5.3.1 SIMP-like method

In the lattice size optimisation described in Section 5.2.1, the optimised lattice structure can have struts with small cross-sectional areas which cannot be manufactured. One way to deal with this issue is to set a minimum manufacturable size as the lower bound for cross-sectional areas of struts. This approach is straightforward but with the sacrifice of keeping redundant struts in the optimised structure which could have been removed. Alternatively, the SIMP approach used in topology optimisation of continuum structures is adapted for lattice structures in order to remove redundant struts which are not critical to the structure. The idea of this proposed SIMP-like lattice topology optimisation method is described as follows.

To begin with, the lattice element stiffness can be expressed equivalently as

$$\mathbf{k}_i = \frac{A_i}{A_i^{\min}} \mathbf{k}_i^{\min} = \rho_i \mathbf{k}_i^{\min} \,, \tag{5.46}$$

where $\mathbf{k}_i^{\min}$ is a reference element stiffness with the prescribed minimum cross-sectional area $A_i^{\min}$, and the areal relative density $\rho_i = A_i / A_i^{\min}$ which is less than 1 when A_i becomes smaller than $A_i^{\min}$.

The SIMP-like approach considers the following modified lattice element stiffness,

$$\mathbf{k}_i^* = \rho_i^* \mathbf{k}_i^{\min} \tag{5.47}$$

with the modified ρ_i^* depending on the value of A_i. If $A_i \geq A_i^{\min}$, $\rho_i^* = \rho_i$; otherwise, a scaling of ρ_i is considered. A similar exponential scaling function as used in the SIMP method can be adopted, see Figure 1.6. The scaling function needs here however to take into account that $\rho_i > 1$ is possible. Hence, the scaling function needs to satisfy that the gradient $\partial \rho_i^* / \partial \rho_i$ at $\rho_i = 1$ is continuous so that in the gradient-based optimisation algorithm the gradient is continuous throughout. An example of a cubic Bézier scaling function satisfying gradient continuity at $\rho_i = 1$ is plotted in Figure 5.17.

The first derivative of the modified lattice element stiffness (5.47) with respect to the cross-sectional area is now computed with

$$\frac{\partial \mathbf{k}_i^*}{\partial A_i} = \frac{\partial \rho_i^*}{\partial A_i} \mathbf{k}_i^{\min} = \frac{\partial \rho_i^*}{\partial \rho_i} \frac{\partial \rho_i}{\partial A_i} \mathbf{k}_i^{\min} = \frac{\partial \rho_i^*}{\partial \rho_i} \frac{\mathbf{k}_i^{\min}}{A_i^{\min}} \,. \tag{5.48}$$

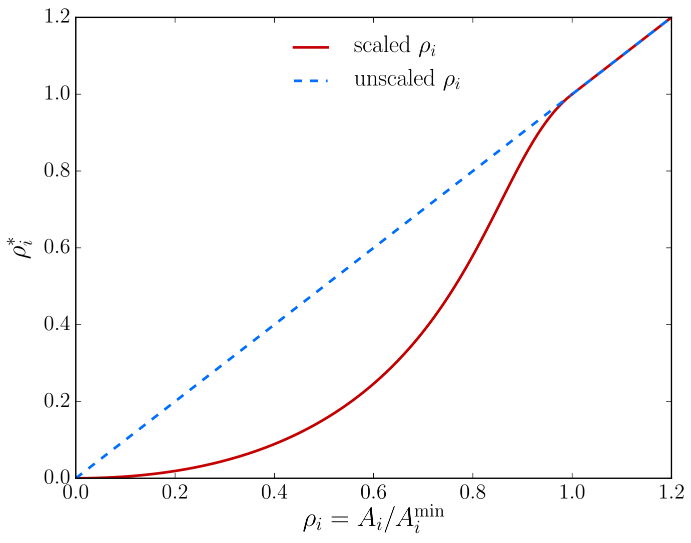


Fig. 5.17 A cubic Bézier scaling function

The gradient of the compliance objective function with the modified lattice element stiffness becomes

$$\frac{\partial J^*}{\partial A_i} = -\mathbf{u}_i^\mathsf{T}(\boldsymbol{A})\frac{\partial \mathbf{k}_i^*(\boldsymbol{A})}{\partial A_i}\mathbf{u}_i(\boldsymbol{A}) = -\mathbf{u}_i^\mathsf{T}(\boldsymbol{A})\frac{\partial \rho_i^*}{\partial \rho_i}\frac{\mathbf{k}_i^{\min}}{A_i^{\min}}\mathbf{u}_i(\boldsymbol{A})\,. \tag{5.49}$$

5.3.2 Sensitivity filtering

As is known in topology optimisation of continuum structures, filtering techniques are used to deal with the chequerboarding problem and other numerical instabilities, including local minima and mesh dependence [61]. For the lattice topology optimisation using the SIMP-like approach introduced as above, lattice struts with small cross-sectional areas can be scattered throughout the optimised structure, which is not desired for the final structure. In order to remove these scattered small struts, a lattice-suitable sensitivity filtering based on lattice unit cells is proposed as follows.

After the gradients of all lattice struts have been obtained with (5.49), the average filtered sensitivity of each lattice unit cell is computed with

$$\overline{\frac{\partial J^*}{\partial A_e}} = \frac{\sum_e w_e \frac{\partial J^*}{\partial A_e}/L_e}{\sum_e w_e/L_e}\,, \tag{5.50}$$

where e represent the lattice strut in a lattice unit cell, L_e is the length of the lattice strut, and w_e is weight given by a linearly decaying distance function which depends

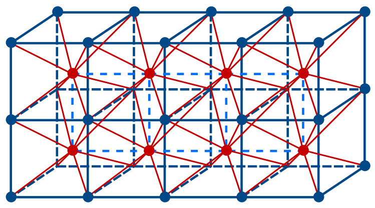


Fig. 5.18 A sketch of BCC lattice unit cell assembly with the red featuring characteristic lattice struts and nodes.

on the centroids of the lattice strut and the lattice unit cell, i.e.

$$w_e(\boldsymbol{x}_e^c, \boldsymbol{x}_{\text{cell}}^c) = \max\left(R_f - ||\boldsymbol{x}_e^c - \boldsymbol{x}_{\text{cell}}^c||, 0\right) , \tag{5.51}$$

where $\boldsymbol{x}_e^c$ and $\boldsymbol{x}_{\text{cell}}^c$ denote the centroid of the lattice strut and the unit cell respectively, and R_f is a user-defined filtering range which should be greater than the size of the unit cell. The motivation of dividing by the lengths of lattice struts in the sensitivity filtering (5.50) is that the term inside the summation can thus be interpreted as the weighted strain energy density, i.e. the strain energy per unit volume.

Figure 5.18 shows a sketch of BCC lattice unit cells. The gradient of each strut in a unit cell is replaced with the average sensitivity of the unit cell. For a lattice strut which is shared by several unit cells, its gradient is computed as the average of the sensitivities of the unit cells that contain the strut. The sensitivity filtering applied in this way ensures that the characteristic lattice struts featuring the same unit cell have the same gradient. A unit cell is removed from the lattice structure if the lattice struts featuring the cell have small cross-sectional areas. As a result, the topology of the lattice structure changes with the removal of unit cells.

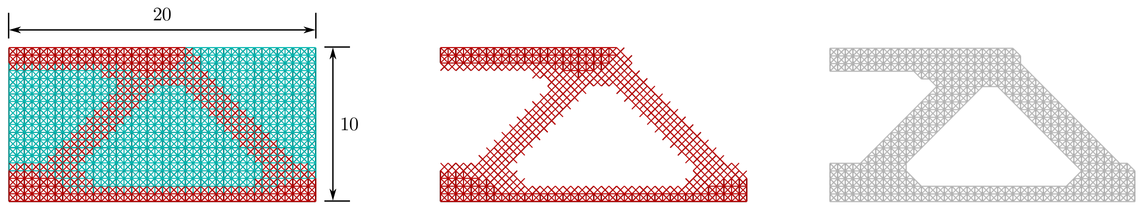
5.4 Examples of lattice topology optimisation

5.4.1 Planar cantilever lattice

A planar cantilever with the dimension 20×10 as shown in Figure 5.19a is considered for lattice topology optimisation. The cantilever lattice structure is comprised of square unit cells with diagonals. A unit force is applied at the right bottom corner of the cantilever. The left side of the cantilever is fixed. The Young's modulus of the material is chosen with $E = 1000$, and the total volume of the lattice structure is 10.

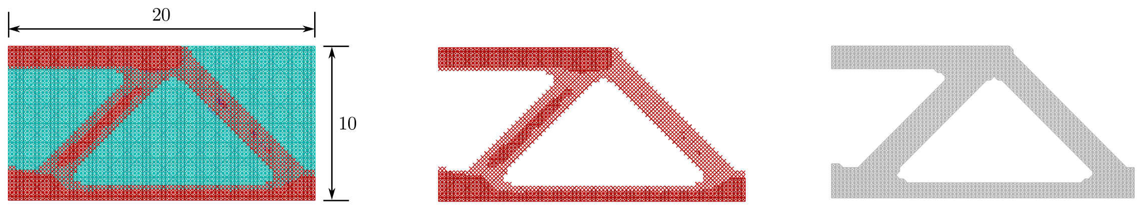
Two different unit cell sizes are considered for the planar cantilever lattice, 0.5×0.5 and 0.25×0.25, as shown in Figure 5.19a and Figure 5.20a, with uniform strut diameters

0.08 and 0.057 respectively. Lattice topology optimisation is performed with the target of achieving 30% of the original lattice volume. The initial cross-sectional areas of the lattice structure are considered as uniform such that the total lattice volume is 3. After removing lattice struts with small cross-sectional areas (e.g. less than 10^{-4} as used in this example), the topology optimised cantilever lattice with the two unit cell sizes are given in Figure 5.19b and Figure 5.20b, respectively, considering the lattice volume ratio 0.3. The final optimised structure is obtained by recovering every unit cell configuration featured by the remaining lattice struts, as shown in Figure 5.19c and Figure 5.20c.



(a) Topology optimisation result (30% of lattice volume) (b) Extracting lattice struts (c) Recovering unit cells

Fig. 5.19 Lattice topology optimisation result with unit cell size 0.5×0.5.



(a) Topology optimisation result (30% of lattice volume) (b) Extracting lattice struts (c) Recovering unit cells

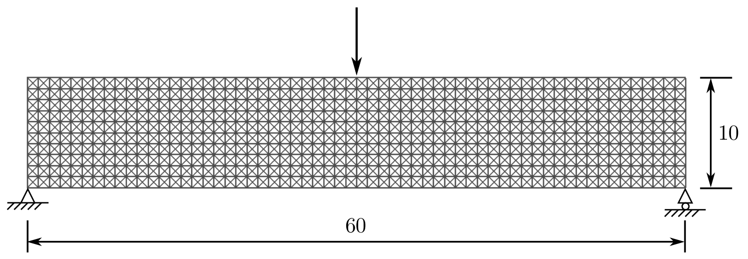
Fig. 5.20 Lattice topology optimisation result with unit cell size 0.25×0.25.

It can be observed from the lattice topology optimisation results that the optimised lattice topologies are similar considering different sizes of unit cells. The compliance of the topology optimised cantilever lattice with unit cell size 0.5×0.5 is 6.64, reduced by 22.1% compared with the initial compliance 8.52 considering the lattice volume 0.3. In contrast, the compliance of the topology optimised cantilever lattice with unit cell size 0.25×0.25 is 6.93, reduced by 23.9% compared with the initial compliance 9.11 considering the lattice volume 0.3.

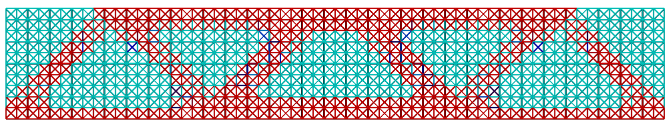
5.4.2 MBB lattice

An MBB lattice structure comprised of square unit cells containing diagonals, as depicted in Figure 5.21a, is considered for lattice topology optimisation. The MBB beam is a benchmark example frequently used in the continuum topology optimisation. The dimension of the MBB lattice structure is 60×10 with the unit cell size 1×1. The lattice structure is simply-supported with a unit point load applied in the upper middle of the structure. The Young's modulus of the material is set as $E = 1000$ and the total volume of the lattice structure is 50. The volume ratio 0.4 is considered for the lattice topology optimisation. The cross-sectional areas of lattice struts are assumed to be uniform in the initial MBB lattice structure. The strut diameters are 0.1465.

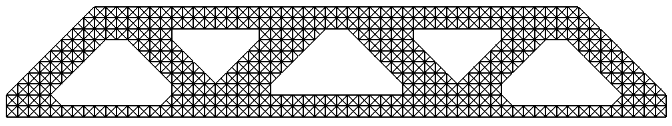
Figure 5.21b shows the topology optimisation result of the MBB lattice structure. After removing lattice struts with small cross-sectional areas (e.g. less than 10^{-4} as used in this example) and recovering unit cells based on the lattice struts remained in the topology optimisation result, the final topology optimised lattice structure is obtained as shown in Figure 5.21c. The compliance of the initial MBB lattice with 40% of the lattice volume is 5.17, and the compliance of the optimised lattice is 3.83, which is reduced by 25.9%.



(a) Initial MBB lattice topology with unit cell size 1×1



(b) Lattice topology optimisation result (40% of lattice volume)



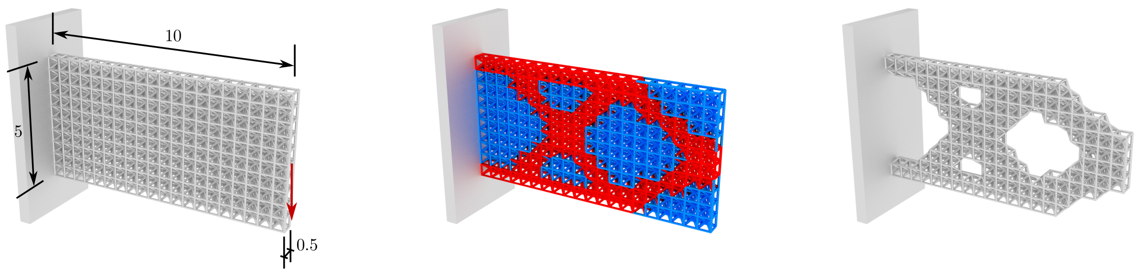
(c) Final topology optimised MBB lattice

Fig. 5.21 Topology optimisation of the MBB lattice structure.

5.4.3 3D cantilever lattice

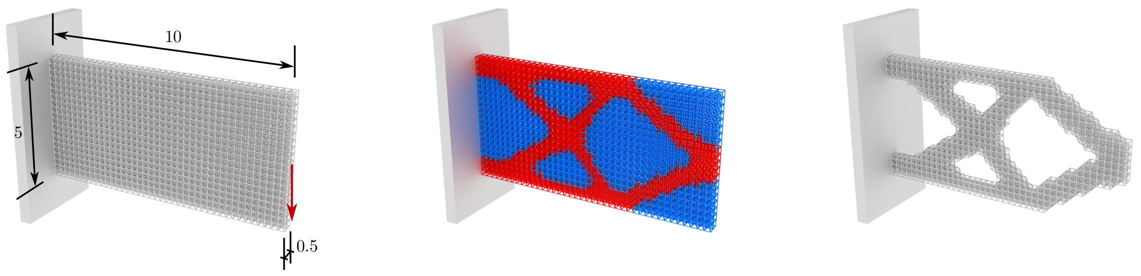
A 3D cantilever lattice structure is considered for lattice topology optimisation. The dimension of the cantilever lattice is $10 \times 5 \times 0.5$. Two different unit cell sizes, 0.25×0.25 and 0.125×0.125 as depicted in Figure 5.22a and Figure 5.23a respectively, are considered. The cantilever lattice is fixed on its left side, and a loading with the magnitude of 100 is applied at the centre on the right side. The Young's modulus of the material is chosen with $E = 10^7$, and the total volume of the lattice is 10. The lattice volume ratio 0.4 is considered for the lattice topology optimisation. The cross-sectional areas of lattice struts are assumed to be uniform in the initial cantilever lattice structure.

The lattice optimisation results are shown in Figure 5.22b and Figure 5.23b for the two different unit cell sizes, respectively. It can be observed that the optimised lattice structures have similar topologies with different unit cell sizes. The compliances of the optimised cantilever lattice are 2.0 for the coarse lattice and 2.1 for the fine lattice.



(a) Initial lattice structure (b) Optimisation result (c) Final optimised lattice

Fig. 5.22 Lattice topology optimisation of the cantilever with unit cell size $0.25 \times 0.25 \times 0.25$ and uniform strut diameter 0.0485.



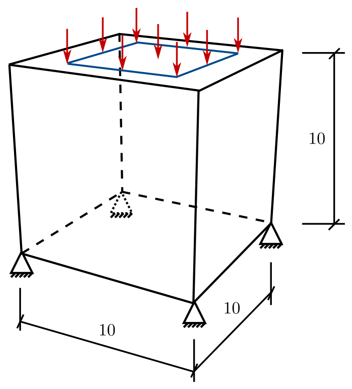
(a) Initial lattice structure (b) Optimisation result (c) Final optimised lattice

Fig. 5.23 Lattice topology optimisation of the cantilever with unit cell size $0.125 \times 0.125 \times 0.125$ and uniform strut diameter 0.0245.

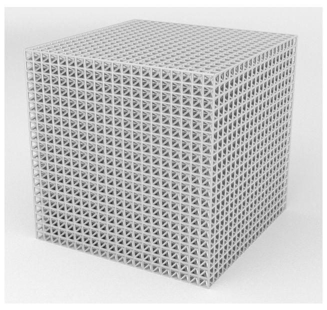
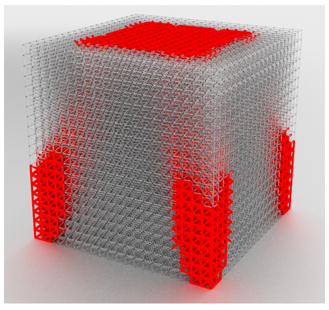
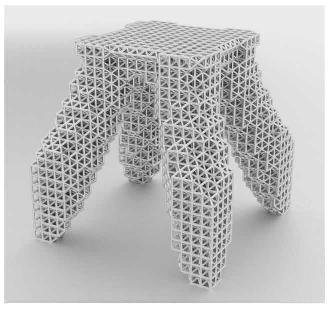
5.4.4 Lattice stool

A lattice stool can be obtained from lattice topology optimisation of a 3D lattice cube as shown in Figure 5.24b. A similar example has been studied using continuum topology optimisation in [19]. The dimension of the lattice cube is $10 \times 10 \times 10$ with the unit cell size $0.5 \times 0.5 \times 0.5$. The cube is fixed at the four bottom corners and is subjected to a pressure with the magnitude of 100 uniformly distributed on an area of size 6×6 on the top, see Figure 5.24a. The total volume of the lattice is 400 with uniform strut diameters 0.0986, and the lattice volume ratio of 0.25 is considered. The Young's modulus of the material is chosen with $E = 10^7$.

The cross-sectional areas of the lattice struts in the initial lattice cube with the lattice volume ratio of 0.25 are assumed to be the same, as shown in Figure 5.24b. Figure 5.24c shows the lattice topology optimisation result. The compliance of the topology optimised lattice cube is 0.0327, reduced by 75.3% compared with that of the initial lattice cube which is 0.132. The final lattice stool shown in Figure 5.24d is extracted from the lattice topology result by removing lattice struts with cross-sectional areas less than 10^{-4} and recovering unit cells from the remaining lattice struts.



(a) Sketch of the lattice cube

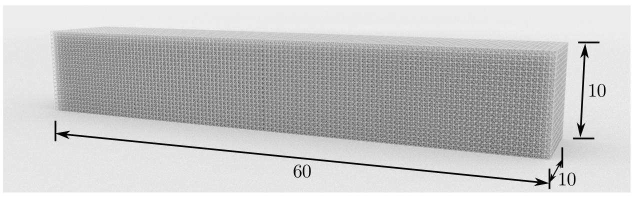
(b) Initial lattice cube (c) Optimisation result (d) Final lattice stool

Fig. 5.24 Lattice topology optimisation of a 3D lattice cube.

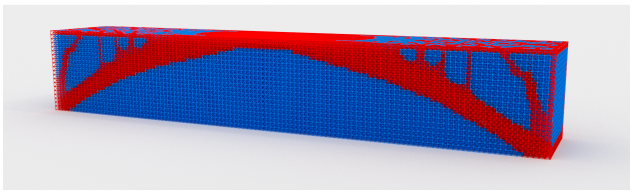
5.4.5 Long-span lattice

Figure 5.25 shows lattice topology optimisation of a long-span lattice structure. The dimension of the initial lattice structure is $60\times10\times10$ with the unit cell size $0.5\times0.5\times0.5$ as shown in Figure 5.25a. A uniform pressure loading with a total magnitude of 200 is applied on the top surface of the lattice structure. The lattice is fixed at the four corners on the bottom. The total lattice volume is 400 with uniform strut diameters 0.04, and the lattice volume ratio 0.2 is considered for the lattice topology optimisation. The cross-sectional areas of the struts in the initial lattice structure are assumed to be the same. The Young's modulus of the material is set as $E = 10^7$.

The topology optimisation result is shown in Figure 5.25b with the lattice volume ratio of 0.2. The compliance of the topology optimised long-span lattice structure is 0.251, reduced by 43.5% compared with that of the initial long-span lattice structure which is 0.444. The final topology optimised long-span lattice structure shown in Figure 5.25c is extracted from the topology optimisation result by removing lattice struts with cross-sectional areas less than 10^{-4} and recovering unit cells from the remaining lattice strut.



(a) Initial long-span lattice structure


(b) Lattice topology optimisation result

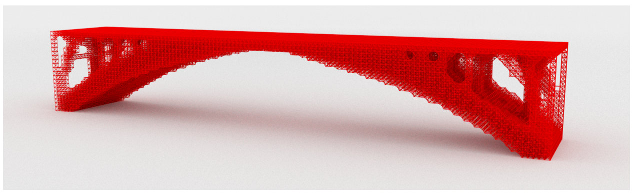

(c) Final optimised long-span lattice structure

Fig. 5.25 Lattice topology optimisation of a 3D long-span lattice structure.

Chapter 6

Optimisation of lattice-skin structures

In this chapter the optimisation of lattice-skin structures, including lattice topology optimisation and shape optimisation, is presented. The SIMP-like regularisation and filtering methods introduced in Section 5.3 are further applied in lattice topology optimisation of lattice-skin structures (Section 6.1). For shape optimisation of lattice-skin structures in Section 6.3, the free-form deformation technique is used to enable the concurrent shape update of the thin-shell and the lattice, and the lattice-skin coupling is also considered in shape optimisation. The sensitivity analysis is conducted in detail in Section 6.1.1 and Section 6.3.2 for lattice topology and shape optimisation of the lattice-skin structure, respectively.

6.1 Lattice-skin topology optimisation

6.1.1 Sensitivity analysis with SIMP-like method

In order to perform the sensitivity analysis of lattice-skin structures, the condensation of dependent degrees of freedom is required for the discretised equilibrium equation system. Recall the discretised equilibrium equation system (4.33), which can be expanded as follows

$$\mathsf{K}^{\mathrm{s}}_{\mathrm{dd}}\mathsf{u}^{\mathrm{s}}_{\mathrm{d}} + \mathsf{K}^{\mathrm{s}}_{\mathrm{dc}}\mathsf{u}^{\mathrm{s}}_{\mathrm{c}} = \mathsf{f}^{\mathrm{s}}_{\mathrm{d}}\,, \tag{6.1a}$$

$$\mathsf{K}^{\mathrm{s}}_{\mathrm{cd}}\mathsf{u}^{\mathrm{s}}_{\mathrm{d}} + \mathsf{K}^{\mathrm{s}}_{\mathrm{cc}}\mathsf{u}^{\mathrm{s}}_{\mathrm{c}} + \mathsf{G}^{\mathsf{T}}_{\mathrm{c}}\boldsymbol{\lambda} = \mathsf{f}^{\mathrm{s}}_{\mathrm{c}}\,, \tag{6.1b}$$

$$\mathsf{K}^{\mathrm{l}}_{\mathrm{dd}}\mathsf{u}^{\mathrm{l}}_{\mathrm{d}} + \mathsf{K}^{\mathrm{l}}_{\mathrm{dc}}\mathsf{u}^{\mathrm{l}}_{\mathrm{c}} = \mathsf{f}^{\mathrm{l}}_{\mathrm{d}}\,, \tag{6.1c}$$

$$\mathbf{K}^{\mathrm{l}}_{\mathrm{cd}}\mathbf{u}^{\mathrm{l}}_{\mathrm{d}} + \mathbf{K}^{\mathrm{l}}_{\mathrm{cc}}\mathbf{u}^{\mathrm{l}}_{\mathrm{c}} - \mathbf{H}^{\mathsf{T}}_{\mathrm{c}}\boldsymbol{\lambda} = \mathbf{f}^{\mathrm{l}}_{\mathrm{c}}\,, \tag{6.1d}$$

$$\mathbf{G}_{\mathrm{c}}\mathbf{u}^{\mathrm{s}}_{\mathrm{c}} - \mathbf{H}_{\mathrm{c}}\mathbf{u}^{\mathrm{l}}_{\mathrm{c}} = \mathbf{0}\,. \tag{6.1e}$$

The variables $\boldsymbol{\lambda}$ and $\mathbf{u}^{\mathrm{l}}_{\mathrm{c}}$ can be eliminated from equations (6.1d) and (6.1e) using static condensation (or computing the Schur complement) [113] with

$$\boldsymbol{\lambda} = \left(\mathbf{H}^{\mathsf{T}}_{\mathrm{c}}\right)^{-1}\left(\mathbf{K}^{\mathrm{l}}_{\mathrm{cd}}\mathbf{u}^{\mathrm{l}}_{\mathrm{d}} + \mathbf{K}^{\mathrm{l}}_{\mathrm{cc}}\mathbf{u}^{\mathrm{l}}_{\mathrm{c}} - \mathbf{f}^{\mathrm{l}}_{\mathrm{c}}\right)\,, \tag{6.2}$$

$$\mathbf{u}^{\mathrm{l}}_{\mathrm{c}} = \mathbf{H}^{-1}_{\mathrm{c}}\mathbf{G}_{\mathrm{c}}\mathbf{u}^{\mathrm{s}}_{\mathrm{c}}\,. \tag{6.3}$$

Substituting these two equations to equations (6.1b) and (6.1c) yields

$$\mathbf{K}^{\mathrm{s}}_{\mathrm{cd}}\mathbf{u}^{\mathrm{s}}_{\mathrm{d}} + \left(\mathbf{K}^{\mathrm{s}}_{\mathrm{cc}} + \mathbf{M}^{\mathsf{T}}\mathbf{K}^{\mathrm{l}}_{\mathrm{cc}}\mathbf{M}\right)\mathbf{u}^{\mathrm{s}}_{\mathrm{c}} + \mathbf{M}^{\mathsf{T}}\mathbf{K}^{\mathrm{l}}_{\mathrm{cd}}\mathbf{u}^{\mathrm{l}}_{\mathrm{d}} = \mathbf{f}^{\mathrm{s}}_{\mathrm{c}} + \mathbf{M}^{\mathsf{T}}\mathbf{f}^{\mathrm{l}}_{\mathrm{c}}\,, \tag{6.4}$$

$$\mathbf{K}^{\mathrm{l}}_{\mathrm{dc}}\mathbf{M}\mathbf{u}^{\mathrm{s}}_{\mathrm{c}} + \mathbf{K}^{\mathrm{l}}_{\mathrm{dd}}\mathbf{u}^{\mathrm{l}}_{\mathrm{d}} = \mathbf{f}^{\mathrm{l}}_{\mathrm{d}}\,, \tag{6.5}$$

with

$$\mathbf{M} = \mathbf{H}^{-1}_{\mathrm{c}}\mathbf{G}_{\mathrm{c}}\,. \tag{6.6}$$

Hence, the discretised equilibrium equation system (4.33) is condensed to

$$\begin{pmatrix} \mathbf{K}^{\mathrm{S}}_{\mathrm{dd}} & \mathbf{K}^{\mathrm{S}}_{\mathrm{dc}} & \mathbf{0} \\ \mathbf{K}^{\mathrm{S}}_{\mathrm{cd}} & \mathbf{K}^{\mathrm{S}}_{\mathrm{cc}} + \mathbf{M}^{\mathsf{T}}\mathbf{K}^{\mathrm{L}}_{\mathrm{cc}}\mathbf{M} & \mathbf{M}^{\mathsf{T}}\mathbf{K}^{\mathrm{L}}_{\mathrm{cd}} \\ \mathbf{0} & \mathbf{K}^{\mathrm{L}}_{\mathrm{dc}}\mathbf{M} & \mathbf{K}^{\mathrm{L}}_{\mathrm{dd}} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{\mathrm{S}}_{\mathrm{d}} \\ \mathbf{u}^{\mathrm{S}}_{\mathrm{c}} \\ \mathbf{u}^{\mathrm{L}}_{\mathrm{d}} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{\mathrm{S}}_{\mathrm{d}} \\ \mathbf{f}^{\mathrm{S}}_{\mathrm{c}} + \mathbf{M}^{\mathsf{T}}\mathbf{f}^{\mathrm{L}}_{\mathrm{c}} \\ \mathbf{f}^{\mathrm{L}}_{\mathrm{d}} \end{pmatrix}\,, \tag{6.7}$$

which can be expressed more compactly as

$$\widetilde{\mathbf{K}}\widetilde{\mathbf{u}} = \widetilde{\mathbf{f}} \tag{6.8}$$

with $\widetilde{\mathbf{K}}$, $\widetilde{\mathbf{u}}$ and $\widetilde{\mathbf{f}}$ denoting the condensed stiffness matrix, the condensed displacement vector and the condensed force vector, respectively.

The compliance of the lattice-skin structure can now be computed as

$$J = \mathbf{f}^{\mathsf{T}}\mathbf{u} = \widetilde{\mathbf{f}}^{\mathsf{T}}\widetilde{\mathbf{u}}\,, \tag{6.9}$$

which leads to the gradient of the compliance with respect to cross-sectional areas of struts, similar to (5.23),

$$\frac{\partial J(\boldsymbol{A})}{\partial A_j} = -\widetilde{\mathbf{u}}^{\mathsf{T}}\frac{\partial\widetilde{\mathbf{K}}}{\partial A_j}\widetilde{\mathbf{u}}\,. \tag{6.10}$$

Since only entries related to the lattice stiffness depend on the cross-sectional areas of struts, (6.10) can be written as follows after substituting (6.3) and (6.7)

$$\frac{\partial J(\boldsymbol{A})}{\partial A_j} = -\left(\mathbf{u}_{\mathrm{c}}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}_{\mathrm{cc}}^{\mathrm{l}}}{\partial A_j}\mathbf{u}_{\mathrm{c}}^{\mathrm{l}} - \left(\mathbf{u}_{\mathrm{d}}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}_{\mathrm{dc}}^{\mathrm{l}}}{\partial A_j}\mathbf{u}_{\mathrm{c}}^{\mathrm{l}} - \left(\mathbf{u}_{\mathrm{c}}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}_{\mathrm{cd}}^{\mathrm{l}}}{\partial A_j}\mathbf{u}_{\mathrm{d}}^{\mathrm{l}} - \left(\mathbf{u}_{\mathrm{d}}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}_{\mathrm{dd}}^{\mathrm{l}}}{\partial A_j}\mathbf{u}_{\mathrm{d}}^{\mathrm{l}}\,, \tag{6.11}$$

which can be written in matrix form as

$$\frac{\partial J(\boldsymbol{A})}{\partial A_j} = -\left(\mathbf{u}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}}{\partial A_j}\mathbf{u}^{\mathrm{l}} \tag{6.12}$$

with

$$\mathbf{u}^{\mathrm{l}} = \begin{pmatrix} \mathbf{u}_{\mathrm{d}}^{\mathrm{l}} \\ \mathbf{u}_{\mathrm{c}}^{\mathrm{l}} \end{pmatrix} \tag{6.13}$$

and

$$\mathbf{K}^{\mathrm{l}} = \begin{pmatrix} \mathbf{K}_{\mathrm{dd}}^{\mathrm{l}} & \mathbf{K}_{\mathrm{dc}}^{\mathrm{l}} \\ \mathbf{K}_{\mathrm{cd}}^{\mathrm{l}} & \mathbf{K}_{\mathrm{cc}}^{\mathrm{l}} \end{pmatrix}\,. \tag{6.14}$$

Considering (5.9), the gradient of compliance can be further written as

$$\frac{\partial J(\boldsymbol{A})}{\partial A_j} = -\left(\mathbf{u}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}}{\partial A_j}\mathbf{u}^{\mathrm{l}} = -\left(\mathbf{u}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}_j^{\mathrm{l}}}{\partial A_j}\mathbf{u}^{\mathrm{l}} = -\left(\mathbf{u}_j^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{k}_j^{\mathrm{l}}}{\partial A_j}\mathbf{u}_j^{\mathrm{l}} \tag{6.15}$$

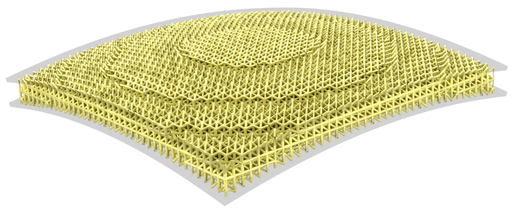
with $\mathbf{k}_j$, $\mathbf{u}_j$ and A_j denoting the element stiffness matrix, the displacement and the cross-sectional area of the j-th strut.

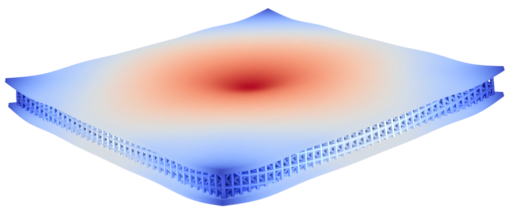
6.2 Examples of lattice-skin topology optimisation

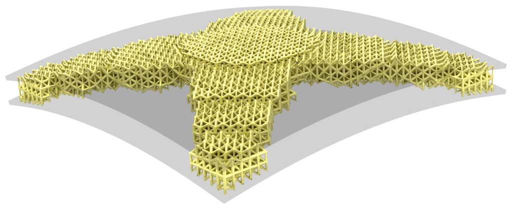
6.2.1 Doubly curved sandwich panel

The doubly curved sandwich panel analysed in Section 4.3.2 is further considered for lattice topology optimisation. All the parameters of the original sandwich panel are the same as before. The total volume of the sandwich panel is 150, and the ratio of the lattice volume to the total material volume is chosen as 0.35 (that is, the lattice volume in the original sandwich panel is 52.5 with uniform diameters 0.0534). The initial lattice topology with BCC unit cells is shown in Figure 6.1a.

The lattice topology of the sandwich panel is optimised considering a lattice volume reduction of 50%. The lattice topology optimisation results are shown in Figure 6.1c and 6.1d. The compliance of the original sandwich panel is 3.75. If the cross-sectional


(a) Original lattice structure


(b) Compliance = 3.75


(c) Topology optimised lattice structure with 50% of lattice volume

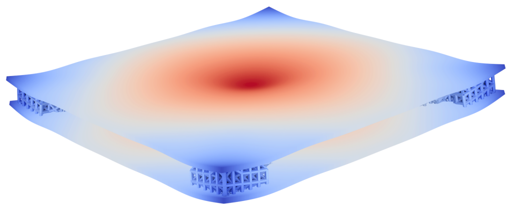

(d) Compliance: initial = 4.92; optimised = 4.2 (50% of lattice volume)

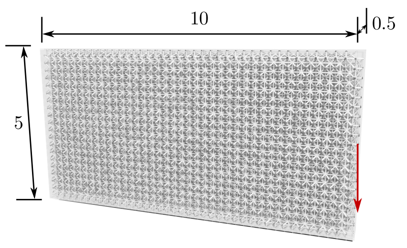
Fig. 6.1 Analysis and topology optimisation of a lattice-skin structure.

areas of struts are reduced uniformly to achieve a 50% volume reduction, that is, the lattice topology remains, the compliance becomes 4.92. In contrast, the compliance of the lattice topology optimised sandwich panel is 4.2, which implies that the lattice volume can be reduced dramatically without compromising the compliance greatly.

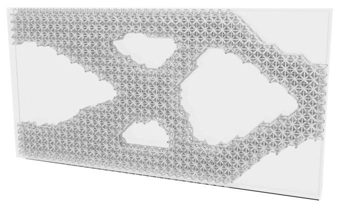
6.2.2 Lattice-infilled cantilever

In order to further examine the effect of lattice volume ratio on the topology optimised lattice-skin structures, a thin-shell cantilever infilled with a BCC lattice is considered, see Figure 6.2a. The dimension of the cantilever is $10 \times 5 \times 0.5$, and the side length of each lattice unit cell is 0.25. A load of the magnitude of 100 is applied at the centre of its right-hand side. The Young's modulus of thin-shell and lattice is chosen as $E = 10^7$. The total volume of the lattice-skin cantilever is prescribed to be 40, and the lattice volume is considered to be reduced by 50% for the lattice topology optimisation.

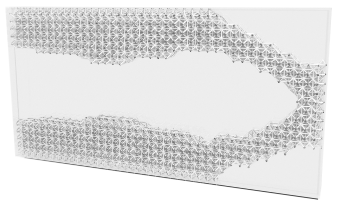
Figure 6.2 shows different lattice topology results considering different initial lattice volume ratios, i.e. different shell thicknesses $\{0.001, 0.01, 0.1\}$. The corresponding strut diameters are 0.088, 0.087 and 0.074, respectively. As the initial lattice volume ratio decreases, the compliance of the corresponding optimised lattice-skin cantilever is reduced as well, indicating again that the lattice structure may not be optimal compared with the solid counterpart with respect to the compliance, as discussed recently in the topology optimisation literature [104].



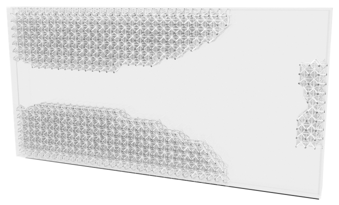
(a) Initial lattice-skin cantilever with unit cell size $0.25 \times 0.25 \times 0.25$



(b) Shell thickness 0.001 with $V_{\text{lattice}}/V_{\text{shell}} = 328.87$; Compliance $= 113.605$



(c) Shell thickness 0.01 with $V_{\text{lattice}}/V_{\text{shell}} = 32$; Compliance $= 82.462$



(d) Shell thickness 0.1 with $V_{\text{lattice}}/V_{\text{shell}} = 2.3$; Compliance $= 25.289$

Fig. 6.2 Lattice topology results considering different lattice volume ratios.

In addition, it is observed that the optimised lattice topology changes as the initial lattice volume ratio is varied. When the initial lattice volume ratio is relatively large, a grid-like topology is obtained (Figure 6.2b), which is similar to the topology optimisation result of the lattice structure shown in Figure 5.23b. However, the grid-like topology disappears as the initial lattice volume ratio is reduced (Figure 6.2c and 6.2d). This can be explained as follows: when the thickness of the thin-shell is very small, the grid-like lattice structure provides the shear rigidity in the height direction; as the thickness of the thin-shell increases, the thin-shell provides the shear rigidity instead, leading to the lattice structure concentrating at the top and bottom regions of the cantilever where the bending stresses are large.

6.3 Lattice-skin shape optimisation

The free-form deformation (FFD) technique provides a flexible tool to modify shapes regardless of the underlying parameterisations, which is a reasonable option for shape optimisation of lattice-skin structures since the thin-shell and the lattice have different parameterisations in the lattice-skin structure. The definition of the free-form deforma-

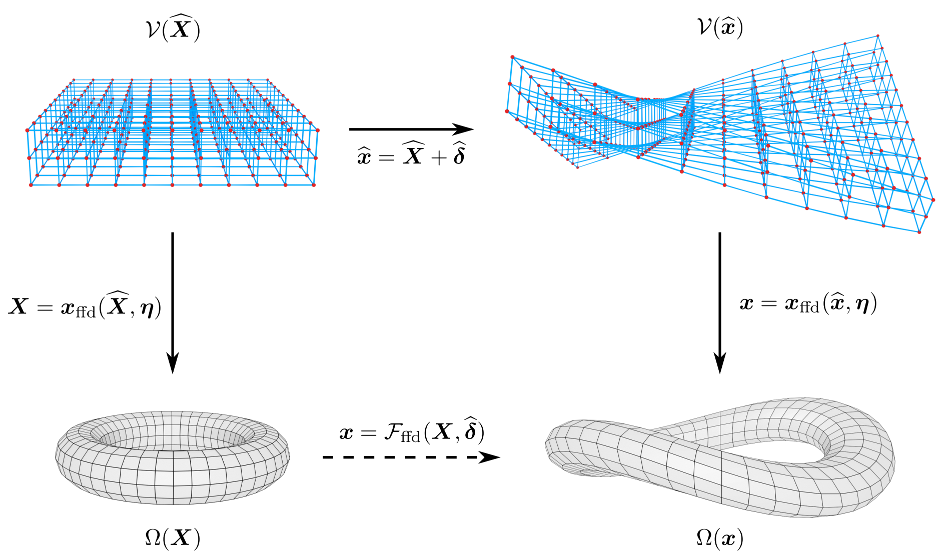

$\mathcal{V}(\widehat{\boldsymbol{X}})$
$\mathcal{V}(\widehat{\boldsymbol{x}})$
$\widehat{\boldsymbol{x}} = \widehat{\boldsymbol{X}} + \widehat{\boldsymbol{\delta}}$
$\boldsymbol{X} = \boldsymbol{x}_{\text{ffd}}(\widehat{\boldsymbol{X}}, \boldsymbol{\eta})$
$\boldsymbol{x} = \boldsymbol{x}_{\text{ffd}}(\widehat{\boldsymbol{x}}, \boldsymbol{\eta})$
$\boldsymbol{x} = \mathcal{F}_{\text{ffd}}(\boldsymbol{X}, \widehat{\boldsymbol{\delta}})$
$\Omega(\boldsymbol{X})$
$\Omega(\boldsymbol{x})$

Fig. 6.3 Free-form deformation map of a torus.

tion is given in Section 6.3.1. The free-form deformation technique is first applied in shape optimisation of thin-shells in order to verify the feasibility of this approach. The sensitivity analysis of the lattice-skin structure using the free-form deformation is then derived in Section 6.3.2, with the consideration of the lattice-skin coupling during the shape optimisation.

6.3.1 Review of free-form deformation

The free-form deformation technique was first proposed by Sederberg et al. [65] for geometric modelling. Given a physical domain Ω which contains the shapes to be deformed, a control volume $\mathcal{V}$ is created with a grid of control points $\widehat{\boldsymbol{X}}$ such that the physical domain is embedded in the control volume. Each material point $\boldsymbol{X}$ in Ω has corresponding parametric coordinates $\boldsymbol{\eta}$ in the parametric domain of the control volume. The parametric coordinates $\boldsymbol{\eta} = (\eta^1, \eta^2, \eta^3)$ are called the free-form coordinates in the following context. The deformation of the physical domain is determined by the control points in the control volume through the free-form coordinates $\boldsymbol{\eta}$. Therefore, the shapes in Ω can be deformed independent of their own parameterisations, which gives great flexibility if the physical domain contains multiple shapes with different parameterisations. Figure 6.3 illustrates the concept of the free-form deformation.

Let the control volume $\mathcal{V} = [v^1_{\text{min}}, v^1_{\text{max}}] \times [v^2_{\text{min}}, v^2_{\text{max}}] \times [v^3_{\text{min}}, v^3_{\text{max}}] \in \mathbb{R}^3$ be defined in terms of trivariate tensor-product Bernstein basis functions with a grid of $(\mu_1 + 1) \times (\mu_2 + 1) \times (\mu_3 + 1)$ control points $\widehat{\boldsymbol{X}}$. The control volume $\mathcal{V}$ is then of degree (μ_1, μ_2, μ_3), and each control point $\widehat{\boldsymbol{X}}_{\boldsymbol{i}}$ with the multi-index $\boldsymbol{i} = \{i_1,\, i_2,\, i_3\}(i_1 = 1, \cdots, \mu_1+1;\ i_2 = 1, \cdots, \mu_2+1;\ i_3 = 1, \cdots, \mu_3+1)$ is given by

$$\widehat{\boldsymbol{X}}_{\boldsymbol{i}} = \widehat{\boldsymbol{X}}_{\boldsymbol{1}} + \frac{i_1-1}{\mu_1}\boldsymbol{v}_1 + \frac{i_2-1}{\mu_2}\boldsymbol{v}_2 + \frac{i_3-1}{\mu_3}\boldsymbol{v}_3\,, \tag{6.16}$$

where $\widehat{\boldsymbol{X}}_{\boldsymbol{1}}$ is a reference point in the control volume $\mathcal{V}$ corresponding to $\boldsymbol{i} = \{1,\, 1,\, 1\}$, and the basis vectors $\boldsymbol{v}_i$ are defined as

$$\boldsymbol{v}_j = (v^j_{\text{max}} - v^j_{\text{min}})\boldsymbol{e}_j \tag{6.17}$$

with $\boldsymbol{e}_j$ $(j = 1,\, 2,\, 3)$ the standard basis for the Euclidean space in the Cartesian coordinate system.

Any material point $\boldsymbol{X}$ in the reference physical domain Ω embedded in the control volume $\mathcal{V}$ is then given by

$$\boldsymbol{X} = \widehat{\boldsymbol{X}}_{\boldsymbol{1}} + \eta^1\boldsymbol{v}_1 + \eta^2\boldsymbol{v}_2 + \eta^3\boldsymbol{v}_3\,, \tag{6.18}$$

where the free-form coordinates $\eta^1, \eta^2, \eta^3 \in [0, 1]$ are determined with

$$\eta^j = (\boldsymbol{X} - \widehat{\boldsymbol{X}}_{\boldsymbol{1}}) \cdot \boldsymbol{v}_j\,. \tag{6.19}$$

The connection between the physical domain Ω and the control volume $\mathcal{V}$ is achieved through the free-form coordinates $\boldsymbol{\eta}$ with

$$\boldsymbol{x}_{\text{ffd}}(\widehat{\boldsymbol{x}}, \boldsymbol{\eta}) = \sum_{\boldsymbol{i}}^{\boldsymbol{\mu}} B^{\boldsymbol{\mu}}_{\boldsymbol{i}}(\boldsymbol{\eta})\widehat{\boldsymbol{x}}_{\boldsymbol{i}} = \sum_{i_1=1}^{\mu_1+1}\sum_{i_2=1}^{\mu_2+1}\sum_{i_3=1}^{\mu_3+1} B^{\mu_1}_{i_1}(\eta^1)B^{\mu_2}_{i_2}(\eta^2)B^{\mu_3}_{i_3}(\eta^3)\widehat{\boldsymbol{x}}_{\boldsymbol{i}} \tag{6.20}$$

with Bernstein polynomials

$$B^{\mu}_{i}(\xi) = \binom{\mu}{i-1}\xi^{i-1}(1-\xi)^{\mu-i+1}\,. \tag{6.21}$$

where $\xi \in [0,\, 1]$. The deformed control points are

$$\widehat{\boldsymbol{x}}_{\boldsymbol{i}} = \widehat{\boldsymbol{X}}_{\boldsymbol{i}} + \widehat{\boldsymbol{\delta}}_{\boldsymbol{i}} \tag{6.22}$$

considering the deformation $\widehat{\boldsymbol{\delta}}_{\boldsymbol{i}}$ of the control volume $\mathcal{V}$.

6.3.2 Sensitivity analysis with free-form deformation

In shape optimisation using the free-form deformation technique, the control points $\widehat{\boldsymbol{x}}$ of the control volume $\mathcal{V}$ of the FFD are the design variables, and the embedded shape is updated accordingly with the optimised control points $\widehat{\boldsymbol{x}}^*$. For the lattice shape in the lattice-skin structure, the lattice nodes which are not coupled with the thin-shell are updated with the optimised control points of the FFD, and the coupled lattice nodes are updated in the way such that they remain coupled with the thin-shell at the same parameters $\boldsymbol{\theta}$.

Sensitivity analysis of thin-shells

The sensitivity of the compliance with respect to the control points $\widehat{\boldsymbol{x}}_{\boldsymbol{i}}$ of the free-form deformation reads

$$\frac{\partial J(\boldsymbol{x}^{\mathrm{s}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \sum_j \frac{\partial J(\boldsymbol{x}^{\mathrm{s}})}{\partial \boldsymbol{x}_j^{\mathrm{s}}(\boldsymbol{\eta})} \frac{\partial \boldsymbol{x}_j^{\mathrm{s}}(\boldsymbol{\eta})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \,, \tag{6.23}$$

where $\boldsymbol{x}_j^{\mathrm{s}}$ denote thin-shell nodes that are influenced by the control point $\widehat{\boldsymbol{x}}_{\boldsymbol{i}}$ of the FFD control volume. The partial derivatives of the compliance with respect to the nodal coordinates of the thin-shell are given by, according to (5.31) derived in Chapter 5,

$$\frac{\partial J(\boldsymbol{x}^{\mathrm{s}})}{\partial \boldsymbol{x}_j^{\mathrm{s}}} = -\sum_e \mathbf{u}^{e\mathsf{T}}(\boldsymbol{x}^{\mathrm{s}}) \frac{\partial \mathbf{k}^e(\boldsymbol{x}^{\mathrm{s}})}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \mathbf{u}^e(\boldsymbol{x}^{\mathrm{s}}) \tag{6.24}$$

with $\mathbf{k}^e$ the element stiffness matrix of the thin-shell, $\mathbf{u}^e$ the nodal displacement vector of the element.

The element stiffness matrix of the thin-shell (4.21) as derived in Chapter 4 is repeated here for convenience,

$$\mathbf{k}_{IJ}^e = \int_{\Omega^e} \left[\frac{Et}{1-\nu^2} \mathbf{M}_I^{\mathsf{T}} \mathbf{H} \mathbf{M}_J + \frac{Et^3}{12(1-\nu^2)} \mathbf{B}_I^{\mathsf{T}} \mathbf{H} \mathbf{B}_J \right] \mathrm{d}\Omega^e \,. \tag{6.25}$$

The first derivatives of the element stiffness matrix (6.25) with respect to the nodal coordinates are computed with

$$\begin{aligned}
\frac{\partial \mathbf{k}_{IJ}^e}{\partial \boldsymbol{x}_j^{\mathrm{s}}} &= \int_{\Omega^e} \frac{Et}{1-\nu^2} \left(\frac{\partial \mathbf{M}_I^{\mathsf{T}}}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \mathbf{H} \mathbf{M}_J + \mathbf{M}_I^{\mathsf{T}} \frac{\partial \mathbf{H}}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \mathbf{M}_J + \mathbf{M}_I \mathbf{H} \frac{\partial \mathbf{M}_J}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \right) \mathrm{d}\Omega^e \\
&+ \int_{\Omega^e} \frac{Et^3}{12(1-\nu^2)} \left(\frac{\partial \mathbf{B}_I^{\mathsf{T}}}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \mathbf{H} \mathbf{B}_J + \mathbf{B}_I^{\mathsf{T}} \frac{\partial \mathbf{H}}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \mathbf{B}_J + \mathbf{B}_I \mathbf{H} \frac{\partial \mathbf{B}_J}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \right) \mathrm{d}\Omega^e \\
&+ \int_{\Omega^e} \left[\frac{Et}{1-\nu^2} \mathbf{M}_I^{\mathsf{T}} \mathbf{H} \mathbf{M}_J + \frac{Et^3}{12(1-\nu^2)} \mathbf{B}_I^{\mathsf{T}} \mathbf{H} \mathbf{B}_J \right] \frac{\partial \mathrm{d}\Omega^e}{\partial \boldsymbol{x}_j^{\mathrm{s}}} \,,
\end{aligned} \tag{6.26}$$

where the partial derivatives of the mid-surface area with respect to the coordinates of thin-shell nodes are computed with

$$\frac{\partial A(\boldsymbol{x}^{\mathrm{s}})}{\partial \boldsymbol{x}^{\mathrm{s}}_j} = \sum_e \int_{\Omega_e} \frac{\partial \mathrm{d}\Omega^e}{\partial \boldsymbol{x}^{\mathrm{s}}_j} = \sum_e \int_{\bar{\Omega}^e} \frac{\partial |\mathcal{J}|}{\partial \boldsymbol{x}^{\mathrm{s}}_j} \, \mathrm{d}\bar{\Omega}^e \tag{6.27}$$

with the Jacobian $|\mathcal{J}| = |\boldsymbol{A}_1 \times \boldsymbol{A}_2|$ and its partial derivatives given in Appendix B.1.

In addition, the second term of the partial derivative in (6.23) is computed with, according to (6.20),

$$\frac{\partial \boldsymbol{x}^{\mathrm{s}}_j(\boldsymbol{\eta})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = B^{\boldsymbol{\mu}}_{\boldsymbol{i}}(\boldsymbol{\eta}) = B^{\mu_1}_{i_1}(\eta^1) B^{\mu_2}_{i_2}(\eta^2) B^{\mu_3}_{i_3}(\eta^3) \, . \tag{6.28}$$

When the mid-surface area is considered as a constraint, the sensitivity of the mid-surface area with respect to the control points $\widehat{\boldsymbol{x}}_{\boldsymbol{i}}$ of the free-form deformation in (6.26) is given by

$$\frac{\partial A(\boldsymbol{x}^{\mathrm{s}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \sum_j \frac{\partial A(\boldsymbol{x}^{\mathrm{s}})}{\partial \boldsymbol{x}^{\mathrm{s}}_j(\boldsymbol{\eta})} \frac{\partial \boldsymbol{x}^{\mathrm{s}}_j(\boldsymbol{\eta})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \, , \tag{6.29}$$

where the area of the mid-surface $A(\boldsymbol{x}^{\mathrm{s}})$ is computed with

$$A(\boldsymbol{x}^{\mathrm{s}}) = \sum_e \int_{\bar{\Omega}^e} |\mathcal{J}| \, \mathrm{d}\bar{\Omega}^e \tag{6.30}$$

with $\bar{\Omega}^e$ the parametric domain of the thin-shell element and $|\mathcal{J}|$ the Jacobian of the mid-surface parameterisation.

Sensitivity analysis of lattice-skin coupling

From the condensed equilibrium equation of the lattice-skin structure (6.7), the sensitivity of the compliance with respect to the control points $\widehat{\boldsymbol{x}}_{\boldsymbol{i}}$ of the FFD is computed with

$$\frac{\partial J(\boldsymbol{x}^{\mathrm{s}}, \boldsymbol{x}^{\mathrm{l}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = -\widetilde{\mathbf{u}}^{\mathsf{T}} \frac{\partial \widetilde{\mathbf{K}}(\boldsymbol{x}^{\mathrm{s}}, \boldsymbol{x}^{\mathrm{l}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \widetilde{\mathbf{u}} \, , \tag{6.31}$$

which yields the following equation after expansion,

$$\begin{aligned} \frac{\partial J(\boldsymbol{x}^{\mathrm{s}}, \boldsymbol{x}^{\mathrm{l}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = &- \left[(\mathbf{u}^{\mathrm{s}}_{\mathrm{c}})^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{s}}_{\mathrm{cc}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{s}}_{\mathrm{c}} + (\mathbf{u}^{\mathrm{s}}_{\mathrm{d}})^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{s}}_{\mathrm{dc}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{s}}_{\mathrm{c}} + (\mathbf{u}^{\mathrm{s}}_{\mathrm{c}})^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{s}}_{\mathrm{cd}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{s}}_{\mathrm{d}} + (\mathbf{u}^{\mathrm{s}}_{\mathrm{d}})^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{s}}_{\mathrm{dd}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{s}}_{\mathrm{d}} \right] \\ &- \left[\left(\mathbf{u}^{l}_{\mathrm{c}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}_{\mathrm{cc}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{l}}_{\mathrm{c}} + \left(\mathbf{u}^{\mathrm{l}}_{\mathrm{d}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}_{\mathrm{dc}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{l}}_{\mathrm{c}} + \left(\mathbf{u}^{\mathrm{l}}_{\mathrm{c}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}_{\mathrm{cd}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{l}}_{\mathrm{d}} + \left(\mathbf{u}^{\mathrm{l}}_{\mathrm{d}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}_{\mathrm{dd}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{l}}_{\mathrm{d}} \right] . \end{aligned} \tag{6.32}$$

The matrix form of (6.32) is expressed as

$$\frac{\partial J(\boldsymbol{x}^{\mathrm{s}}, \boldsymbol{x}^{\mathrm{l}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = -\left(\mathbf{u}^{\mathrm{s}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{s}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{s}} - \left(\mathbf{u}^{\mathrm{l}}\right)^{\mathsf{T}} \frac{\partial \mathbf{K}^{\mathrm{l}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{\mathrm{l}} \tag{6.33}$$

with

$$\mathbf{u}^{\mathrm{s}} = \begin{pmatrix} \mathbf{u}^{\mathrm{s}}_{\mathrm{d}} \\ \mathbf{u}^{\mathrm{s}}_{\mathrm{c}} \end{pmatrix}, \quad \mathbf{u}^{\mathrm{l}} = \begin{pmatrix} \mathbf{u}^{\mathrm{l}}_{\mathrm{d}} \\ \mathbf{u}^{\mathrm{l}}_{\mathrm{c}} \end{pmatrix} \tag{6.34}$$

and

$$\mathbf{K}^{\mathrm{s}} = \begin{pmatrix} \mathbf{K}^{\mathrm{s}}_{\mathrm{dd}} & \mathbf{K}^{\mathrm{s}}_{\mathrm{dc}} \\ \mathbf{K}^{\mathrm{s}}_{\mathrm{cd}} & \mathbf{K}^{\mathrm{s}}_{\mathrm{cc}} \end{pmatrix}, \quad \mathbf{K}^{\mathrm{l}} = \begin{pmatrix} \mathbf{K}^{\mathrm{l}}_{\mathrm{dd}} & \mathbf{K}^{\mathrm{l}}_{\mathrm{dc}} \\ \mathbf{K}^{\mathrm{l}}_{\mathrm{cd}} & \mathbf{K}^{\mathrm{l}}_{\mathrm{cc}} \end{pmatrix}. \tag{6.35}$$

The expression (6.33) can be further written as follows, according to (5.31),

$$\frac{\partial J(\boldsymbol{x}^{\mathrm{s}}, \boldsymbol{x}^{\mathrm{l}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = -\sum_{e_{\mathrm{s}}} \left(\mathbf{u}^{e_{\mathrm{s}}}\right)^{\mathsf{T}} \frac{\partial \mathbf{k}^{e_{\mathrm{s}}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{e_{\mathrm{s}}} - \sum_{e_{\mathrm{l}}} \left(\mathbf{u}^{e_{\mathrm{l}}}\right)^{\mathsf{T}} \frac{\partial \mathbf{k}^{e_{\mathrm{l}}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{e_{\mathrm{l}}}, \tag{6.36}$$

where e_{s} and e_{l} denote the thin-shell element and the lattice element respectively; $\mathbf{k}^{e_{\mathrm{s}}}$ and $\mathbf{k}^{e_{\mathrm{l}}}$ denote the element stiffness matrix of the thin-shell and the lattice; $\mathbf{u}^{e_{\mathrm{s}}}$ and $\mathbf{u}^{e_{\mathrm{l}}}$ denote the element displacement vector of the thin-shell and the lattice.

The partial derivatives of the element stiffness matrix of the thin-shell in (6.36) are given by

$$\frac{\partial \mathbf{k}^{e_{\mathrm{s}}}(\boldsymbol{x}^{\mathrm{s}})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \sum_{j} \frac{\partial \mathbf{k}^{e_{\mathrm{s}}}(\boldsymbol{x}^{\mathrm{s}})}{\partial \boldsymbol{x}^{\mathrm{s}}_{j}(\boldsymbol{\eta})} \frac{\partial \boldsymbol{x}^{\mathrm{s}}_{j}(\boldsymbol{\eta})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \tag{6.37}$$

which has been derived in Section 6.3.2.

For the lattice structure, non-coupled lattice nodes are updated according to the control points of the FFD control volume, and the coupled lattice nodes are moved with the updated thin-shell surface to couple at the same parameters. Hence, the second term on the right-hand side of (6.36) is expressed as

$$\sum_{e_{\mathrm{l}}} \left(\mathbf{u}^{e_{\mathrm{l}}}\right)^{\mathsf{T}} \frac{\partial \mathbf{k}^{e_{\mathrm{l}}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{e_{\mathrm{l}}} = \sum_{e_{\mathrm{c}}} \left(\mathbf{u}^{e_{\mathrm{c}}}\right)^{\mathsf{T}} \frac{\partial \mathbf{k}^{e_{\mathrm{c}}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{e_{\mathrm{c}}} + \sum_{e_{\mathrm{d}}} \left(\mathbf{u}^{e_{\mathrm{d}}}\right)^{\mathsf{T}} \frac{\partial \mathbf{k}^{e_{\mathrm{d}}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \mathbf{u}^{e_{\mathrm{d}}}, \tag{6.38}$$

where e_{c} denotes lattice elements containing coupling lattice nodes, and e_{d} denotes lattice elements without coupling lattice nodes; the first derivatives of lattice element stiffness $\mathbf{k}^{e_{\mathrm{c}}}$ and $\mathbf{k}^{e_{\mathrm{d}}}$ are computed with, respectively,

$$\frac{\partial \mathbf{k}^{e_{\mathrm{c}}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \sum_{j} \sum_{k} \frac{\partial \mathbf{k}^{e_{\mathrm{c}}}(\boldsymbol{x}^{\mathrm{l}})}{\partial \boldsymbol{x}^{\mathrm{l}}_{j}} \frac{\partial \boldsymbol{x}^{\mathrm{l}}_{j}}{\partial \boldsymbol{x}^{\mathrm{s}}_{k}} \frac{\partial \boldsymbol{x}^{\mathrm{s}}_{k}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \tag{6.39}$$

and

$$\frac{\partial \mathbf{k}^{e_\mathrm{d}}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \sum_j \frac{\partial \mathbf{k}^{e_\mathrm{d}}(\boldsymbol{x}^l)}{\partial \boldsymbol{x}_j^\mathrm{l}} \frac{\partial \boldsymbol{x}_j^\mathrm{l}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \,. \tag{6.40}$$

The derivatives of coordinates of coupled lattice nodes with respect to coordinates of thin-shell nodes are computed with

$$\frac{\partial \boldsymbol{x}_j^\mathrm{l}(\theta^1,\theta^2)}{\partial \boldsymbol{x}_k^\mathrm{s}} = N_k(\theta^1,\theta^2)\,, \tag{6.41}$$

where $N_k(\theta^1,\theta^2)$ is the shape function value corresponding to the k-th thin-shell node evaluated at the parameters (θ^1,θ^2) where the j-th lattice node is coupled with the thin-shell.

When the total volume of the lattice-skin structure is considered as a constraint, the sensitivity of the volume of the lattice-skin structure is computed with

$$\frac{\partial V(\boldsymbol{x}^\mathrm{s},\boldsymbol{x}^\mathrm{l})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \frac{\partial V^\mathrm{s}(\boldsymbol{x}^\mathrm{s})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} + \frac{\partial V^\mathrm{l}(\boldsymbol{x}^\mathrm{l})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \frac{\partial A^\mathrm{s}(\boldsymbol{x}^\mathrm{s})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} t^\mathrm{s} + \sum_{e_\mathrm{l}} A^{e_\mathrm{l}} \frac{\partial L^{e_\mathrm{l}}(\boldsymbol{x}^\mathrm{l})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} \,, \tag{6.42}$$

where t^s is the thickness of the thin-shell, A^{e_l} is the cross-sectional area of a strut, and L^{e_l} is the length of the strut. The partial derivatives of the thin-shell area are given by (6.29), and the partial derivatives of the strut length L^{e_l} are computed with

$$\frac{\partial L^{e_\mathrm{l}}(\boldsymbol{x}^\mathrm{l})}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}} = \begin{cases} \displaystyle\sum_j \sum_k \frac{\partial L^{e_\mathrm{l}}(\boldsymbol{x}^\mathrm{l})}{\partial \boldsymbol{x}_j^\mathrm{l}} \frac{\partial \boldsymbol{x}_j^\mathrm{l}}{\partial \boldsymbol{x}_k^\mathrm{s}} \frac{\partial \boldsymbol{x}_k^\mathrm{s}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}}\,, & \text{if any coupling nodes are contained;} \\ \displaystyle\sum_j \frac{\partial L^{e_\mathrm{l}}(\boldsymbol{x}^\mathrm{l})}{\partial \boldsymbol{x}_j^\mathrm{l}} \frac{\partial \boldsymbol{x}_j^\mathrm{l}}{\partial \widehat{\boldsymbol{x}}_{\boldsymbol{i}}}\,, & \text{otherwise.} \end{cases} \tag{6.43}$$

6.4 Examples of lattice-skin shape optimisation

6.4.1 Square thin-shell

A square thin-shell of the dimension $20\,\mathrm{m} \times 20\,\mathrm{m}$ under a uniform pressure of $100\,\mathrm{kN/m^2}$ is considered, as shown in Figure 6.4a. The Young's modulus $E = 70\,\mathrm{GPa}$ and the Poisson's ratio $\nu = 0.35$. The thickness of the thin-shell is $0.2\,\mathrm{m}$. The displacement of the square thin-shell is shown in Figure 6.4b with the maximum displacement of $72.6\,\mathrm{mm}$ at the middle points of the four edges, and the structural compliance is $1666.24\,\mathrm{kN \cdot m}$. For shape optimisation of the thin-shell, the mid-surface area is prescribed, that is, the area of the optimised shape should not be 1.2 times larger than the initial mid-surface area.

Figure 6.4c shows the optimised shape when the coordinates of shell nodes are taken directly as design variables in shape optimisation. The optimised structural compliance is $173.477\,\mathrm{kN}\cdot\mathrm{m}$, reduced by $89.6\,\%$ compared with the initial square thin-shell. However, the optimised shape is not reasonable for practical use as the shape presents many irregular ripples. As a matter of fact, taking nodal coordinates in a continuum structure as design variables is not a good practice in general since the shape would be severely distorted and the optimisation process would be stuck in local optimal [64].

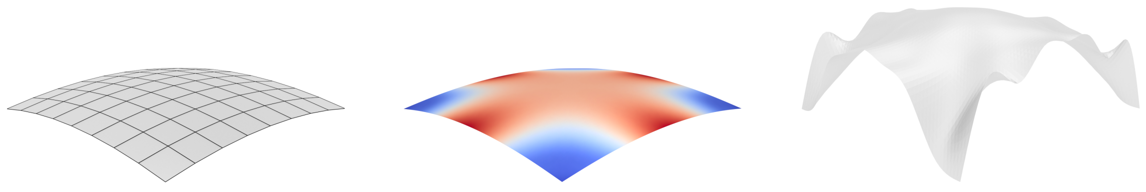

(a) Control mesh (b) Maximum disp. $= 72.6\,\mathrm{mm}$ (c) Optimised shape

Fig. 6.4 Shell node based shape optimisation of the square thin-shell.

In contrast, the optimised shape obtained with the free-form deformation method is promising, as shown in Figure 6.5a. The degrees of the free-form deformation control volume used for the optimisation is $5 \times 5 \times 2$. The optimised structural compliance is $164.015\,\mathrm{kN}\cdot\mathrm{m}$, reduced by 90.2% compared with the initial square thin-shell and also less than the compliance obtained by optimising the nodal coordinates directly. The limit surface of the optimised shell by the free-form deformation is shown in Figure 6.5b, which is apparently more reasonable for practical use compared with the shape shown in Figure 6.4c.

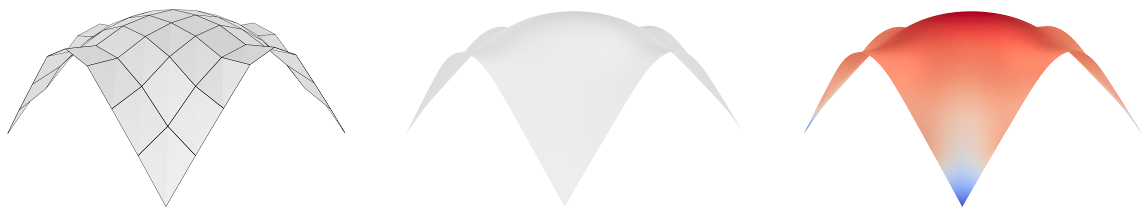
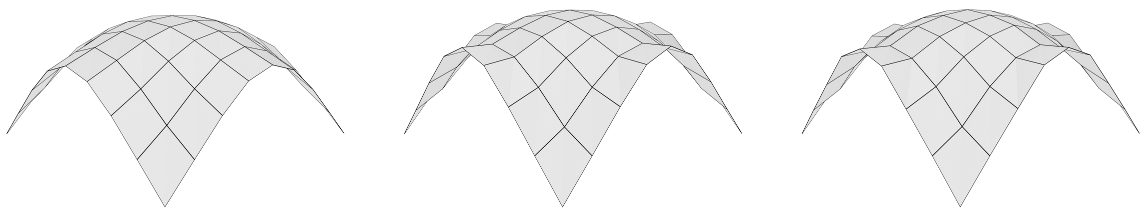

(a) FFD optimised shape (b) Limit optimised surface (c) Maximum disp. $= 4.98\,\mathrm{mm}$

Fig. 6.5 Shape optimisation of the square thin-shell with FFD.

The influence of the number of degrees of the FFD control volume has been investigated as shown in Figure 6.6. The shape cannot be fully optimised with a small number of degrees of the FFD control volume (Figure 6.6a) as the real optimised shape would be highly curved and using fewer number of degrees of the FFD control volume cannot reveal the information of curvature completely. In addition, as the degrees of the FFD control volume increase, the optimised shape varies slightly and the compliance is reduced with a negligible amount (see Figure 6.6b and Figure 6.6c), while it takes more iterations to converge. Therefore, it is sensible to select a moderate number of degrees of the FFD control volume to obtain a reasonable optimisation result. In this example, $5 \times 5 \times 2$ is an appropriate choice for the degrees of the FFD control volume.


(a) FFD: $3 \times 3 \times 2$
Comp. $= 165.59\,\mathrm{kN \cdot m}$

(b) FFD: $5 \times 5 \times 2$
Comp. $= 164.015\,\mathrm{kN \cdot m}$

(c) FFD: $10 \times 10 \times 2$
Comp. $= 163.914\,\mathrm{kN \cdot m}$

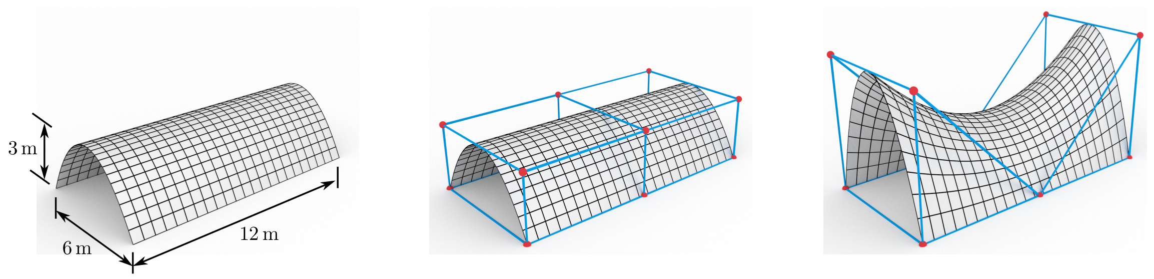
Fig. 6.6 Optimised shapes and compliances using different number of degrees for the FFD control volume.

6.4.2 Parabolic thin-shell

Next the parabolic thin-shell shown in Figure 6.7a with the rectangular plan form $12\,\mathrm{m} \times 6\,\mathrm{m}$ and the height $3\,\mathrm{m}$ is considered. The Young's modulus and the Poisson's ratio are chosen with $E = 30\,\mathrm{GPa}$ and $\nu = 0.2$ respectively. The uniform pressure loading is $5\,\mathrm{kN/m^2}$. The parabolic thin-shell is simply-supported along the two short edges.

An optimised shape of the parabolic thin-shell considering the same conditions is given in [114]. In order to obtain that shape, the thin-shell is constrained to be doubly symmetric to preserve the parabolic shape in the optimisation process. In addition, the maximum height is restricted not to exceed $6\,\mathrm{m}$. With these geometry constraints, the final optimised parabolic shape obtained in [114] has the height $6\,\mathrm{m}$ at the two ends and the height $3.12\,\mathrm{m}$ at the centre.

In order to compare with the optimisation result given by [114], the control volume of the free-form deformation of the shell is created such that the parabolic shape can be



(a) The parabolic shell

(b) FFD control volume
Comp. $= 2.59\,\mathrm{kN} \cdot \mathrm{m}$

(c) Shape optimised shell
Comp. $= 0.183\,\mathrm{kN} \cdot \mathrm{m}$

Fig. 6.7 Shape optimisation of the parabolic shell using a FFD control volume of degrees $2 \times 1 \times 1$.

best approximated. Therefore, the degrees of the FFD control volume are considered as $2 \times 1 \times 1$, that is, linear in directions of the short span and the height and quadratic in the long span direction, as shown in Figure 6.7b. The structural compliance of the initial parabolic thin-shell is $2.59\,\mathrm{kN} \cdot \mathrm{m}$, and the compliance of the optimised thin-shell is $0.183\,\mathrm{kN} \cdot \mathrm{m}$, which is reduced by 92.9%. The optimised shape is shown in Figure 6.7c and has the height 6 m at the two ends and 3 m at the centre, which agrees very well with the optimisation result given by [114]. Figure 6.8 shows the compliance reduction during the optimisation iterations.

As investigated in [20] where the multiresolution optimisation scheme is applied, the optimised shape of the parabolic shell depends on the the initial meshes and the resolutions used. Hence, three different FFD control volumes with varying resolutions (i.e. the degrees of the control volume) have been chosen in order to investigate their influence on shape optimisation. The thin-shell is optimised with the constraint that the mid-surface area is the same for the three different optimised shapes, which is the area of the optimised parabolic shape obtained previously, see Figure 6.7c. The optimised shapes and compliances with the three different degrees of the FFD are shown in Figure 6.9. As can be seen from the results, the parabolic shape obtained with the FFD degree $2 \times 1 \times 1$ is actually not the optimal due to the constraint of keeping the approximately parabolic shape. The compliance of the optimised shape is reduced as the degree of FFD increases, while it takes a longer time to converge for higher degrees of FFD. In addition, more details of the surface curvature can be revealed in the optimised shape when higher degrees of the free-form deformation are used since the shape can deform with more degrees of freedom.

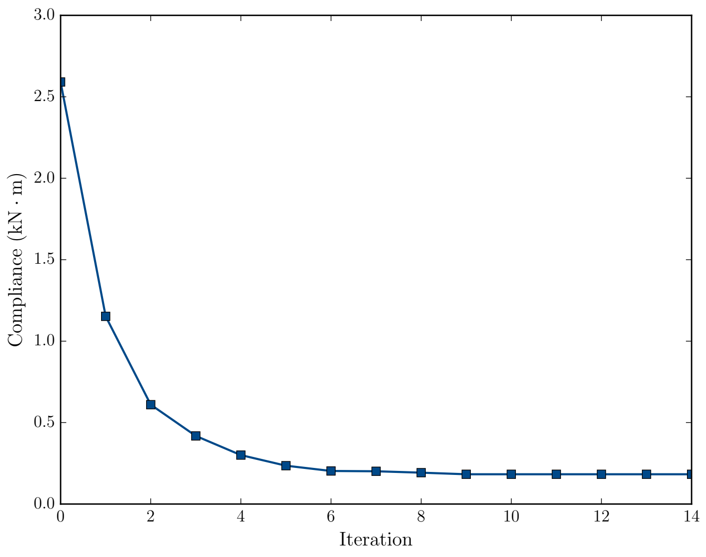
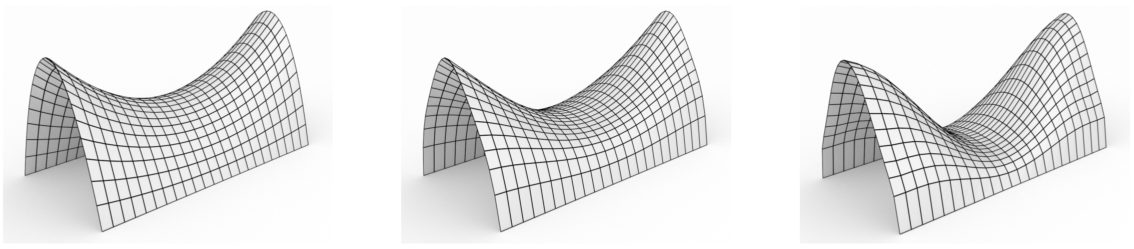


Fig. 6.8 Convergence of the compliance of the parabolic shell.



(a) FFD: $2 \times 1 \times 1$
Comp. $= 0.183\,\mathrm{kN \cdot m}$

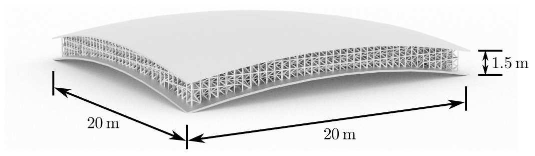
(b) FFD: $4 \times 2 \times 2$
Comp. $= 0.0782\,\mathrm{kN \cdot m}$

(c) FFD: $8 \times 4 \times 4$
Comp. $= 0.0676\,\mathrm{kN \cdot m}$

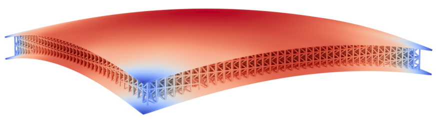
Fig. 6.9 Optimised shapes and compliances using different degrees of FFD control volumes.

6.4.3 Doubly curved sandwich panel

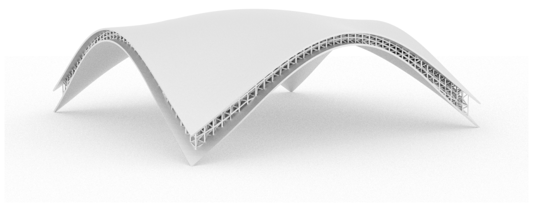
The doubly curved sandwich panel with a square plan of size $20\,\mathrm{m} \times 20\,\mathrm{m}$ is shown in Figure 6.10a. The distance between the top and bottom thin-shells is 1.5 m. The total volume of the sandwich panel is $150\,\mathrm{m}^3$ with the lattice volume ratio of 0.35. Specifically, the shell thickness is about 0.12 m and the strut diameter is about 0.053 m. Each lattice unit cell has a side length of 0.5 m. The Young's modulus and the Poisson's ratio of the material are $E = 70\,\mathrm{GPa}$ and $\nu = 0.35$, respectively. The corners of the sandwich panel are simply-supported, and the upper thin-shell of the panel is subjected to a uniform pressure loading of $100\,\mathrm{kN/m^2}$.



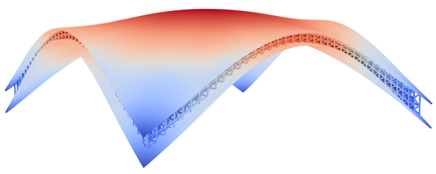
(a) The sandwich panel



(b) Compliance = $931.3\,\mathrm{kN}\cdot\mathrm{m}$



(c) Shape optimised sandwich panel



(d) Compliance = $183.45\,\mathrm{kN}\cdot\mathrm{m}$

Fig. 6.10 Shape optimisation of the sandwich panel with FFD.

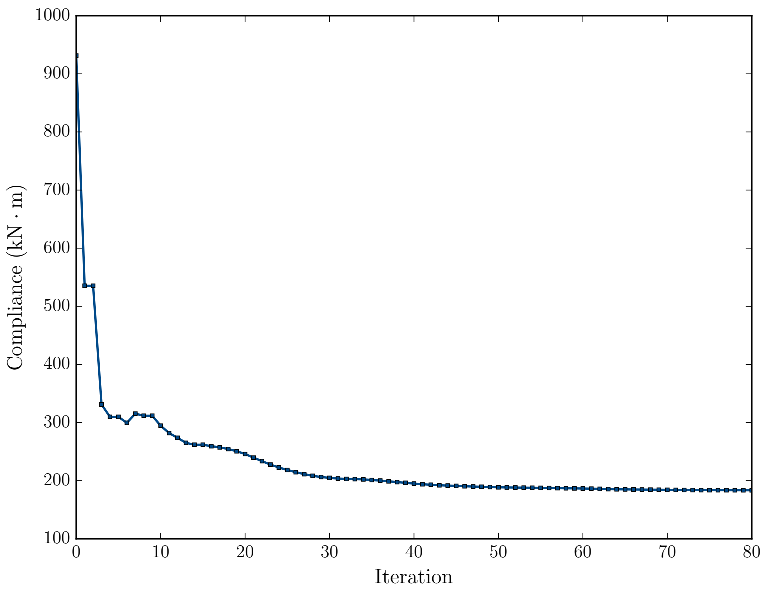
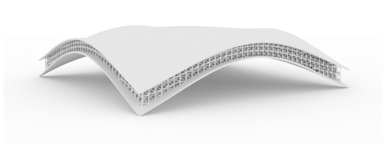
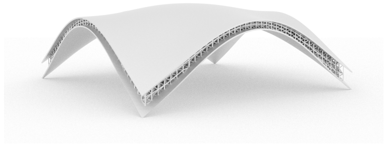


Fig. 6.11 Convergence of the compliance of the doubly curved sandwich panel.

The structural compliance of the initial sandwich panel is $931.3\,\mathrm{kN}\cdot\mathrm{m}$ (Figure 6.10b). Shape optimisation was performed with a FFD control volume containing $6\times6\times3$ nodes. A volume constraint is considered such that the total material volume, proportional to


(a) Lattice volume ratio 0.15: initial 830.279 kN · m; optimised 303.945 kN · m


(b) Lattice volume ratio 0.35: initial 931.3 kN · m; optimised 183.45 kN · m

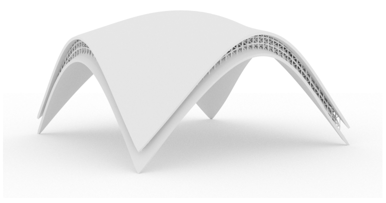

(c) Lattice volume ratio 0.7: initial 1381.02 kN · m; optimised 241.15 kN · m

Fig. 6.12 Shape optimisation considering different lattice volume ratios.

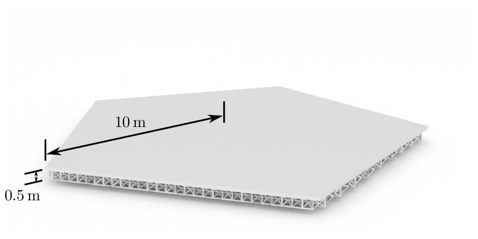
the mass, of the shape optimised panel is the same as the original sandwich panel. In the optimisation process, the shell thickness and the strut diameters remain constant. The shape optimised sandwich panel is shown in Figure 6.10c. The compliance of the shape optimised panel is 183.45 kN · m, reduced by 80.3% compared with the compliance of the original panel while maintaining the same total material volume. The convergence of the optimisation iterations is shown in Figure 6.11.

The example in Chapter 4 indicates that the lattice volume ratio has an influence on the structural compliance. In order to investigate the effect of the lattice volume ratio on the shape optimisation, three different lattice volume ratios, 0.15, 0.35 and 0.7, are considered with uniform diameters 0.035 m, 0.053 m and 0.075 m, respectively. The shape optimisation results are shown in Figure 6.12. The total material volume of the sandwich panel is the same in the three cases. It can be observed that the height of the sandwich panel becomes higher with a larger lattice volume ratio. This is because when the lattice volume ratio is large there would be more lattice volume reduced and compensated with the increase of shell volume. When the lattice volume ratio is small (for example 0.15 as considered in Figure 6.12a), the compliance of the shape optimised panel is larger than the other two cases. The reason can be that as the lattice volume decreases in the optimised shape the coupling effect between the top and the bottom shells is weakened, leading to a reduced overall stiffness.

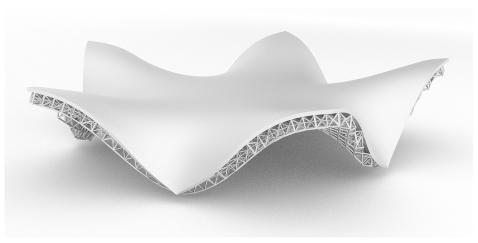
6.4.4 Pentagon roof

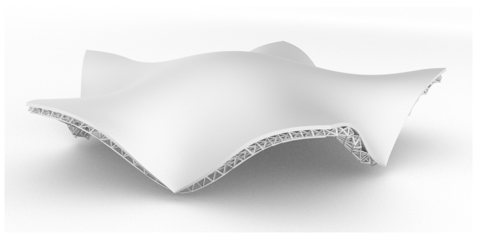
A pentagon roof is studied to show the feasibility of combining shape and lattice topology optimisation to design architectural lattice-skin structures. The radius of the circumcircle of the regular pentagon is 10 m. The Young's modulus and the Poisson's ratio are $E = 70\,\mathrm{GPa}$ and $\nu = 0.35$. The magnitude of the uniform pressure loading is $100\,\mathrm{kN/m^2}$. The thickness of the thin-shell is 0.2 m, and the lattice struts have the same initial cross-sectional areas with diameters 0.1 m. Each lattice unit cell has a side length of 0.5 m.

The lattice structure was first designed for a plate with a regular pentagon shape as shown in Figure 6.13a. The displacements of the uniformly loaded plate are used to generate the initial roof geometry shown in Figure 6.13b. For the roof structure obtained in this way, the lattice is guaranteed to be conformal to the shell shape since the deformation obtained with the finite element analysis ensures the coupling of lattice nodes and the thin-shell. This approach resembles the hanging chain or cloth models used in form-finding in architectural engineering, which, however, may not yield the optimal shape. Therefore, shape optimisation and lattice topology optimisation have been performed to obtain an optimised pentagon lattice shell roof.



(a) Pentagon plate


(b) Initial pentagon roof
Comp. $= 55.12\,\mathrm{kN \cdot m}$


(c) Shape optimised pentagon roof
Comp. $= 35.56\,\mathrm{kN \cdot m}$

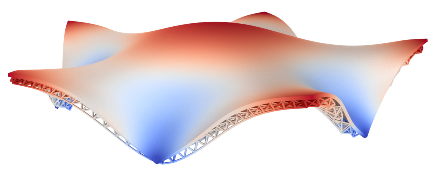
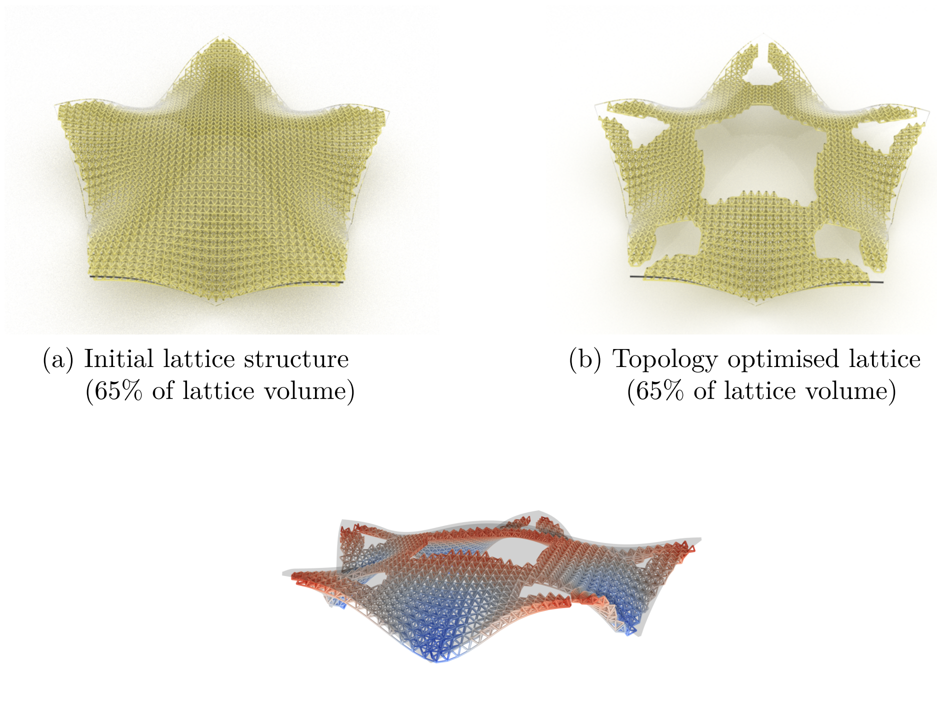

(d) Displacement under uniform loading

Fig. 6.13 Shape optimisation of the pentagon roof.


(a) Initial lattice structure (65% of lattice volume)

(b) Topology optimised lattice (65% of lattice volume)

(c) Deformation under uniform loading

Fig. 6.14 Lattice topology optimisation of the pentagon roof.

The structural compliance of the initial pentagon roof shown in Figure 6.13b is 55.12 kN · m under the prescribed pressure loading. Figure 6.13c shows the shape optimised pentagon roof structure. A volume constraint is considered in the shape optimisation, that is, the optimised roof has the same total volume as the initial roof. The compliance of this optimised pentagon roof is 35.56 kN · m, reduced by 35.5% compared with the initial compliance. The displacement of the shape optimised pentagon roof under the uniform pressure loading is shown in Figure 6.13d.

For the shape optimised lattice shell roof, there are some redundant lattice struts which do not contribute much to the stiffness of the structure and could have been removed. Therefore, lattice topology optimisation was performed subsequently on the shape optimised pentagon roof for the sake of reducing the lattice volume while not compromising the structural stiffness greatly. Another benefit of lattice topology optimisation lies in the architectural aspect since the resulting openings in the lattice structure allow a better lighting condition for the indoor environment.

The aim of lattice topology optimisation considered is to reduce the lattice volume by 35% compared with the lattice volume in the shape optimised pentagon roof

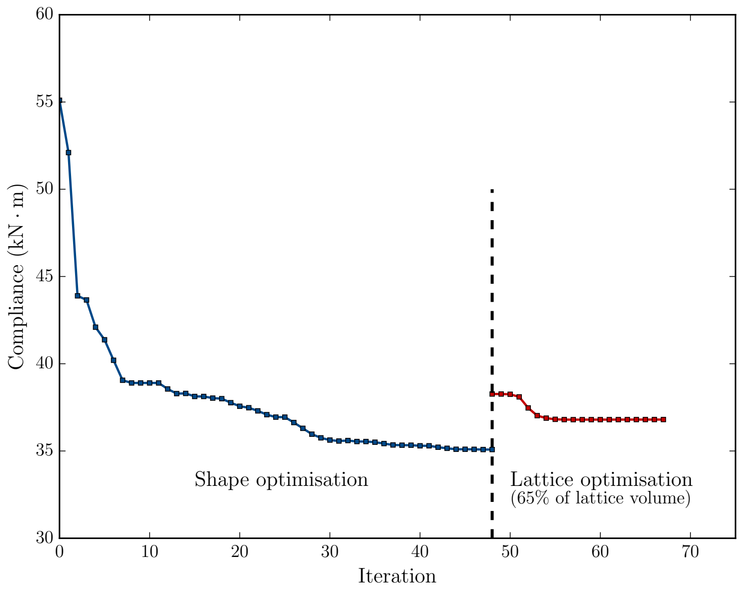


Fig. 6.15 Convergence of shape and lattice topology optimisation.

(Figure 6.13c). The initial lattice structure used for lattice topology optimisation consists of struts with cross-sectional areas reduced uniformly to achieve the 35% reduction of lattice volume, as shown in Figure 6.14a, while the lattice topology remains the same as the shape optimised roof. The structural compliance of this pentagon roof with 65% of lattice material volume is 38.27 kN · m, greater than that of the shape optimised roof as expected since the lattice volume is reduced.

The lattice topology result is shown in Figure 6.14b. Figure 6.14c shows the displacement of the optimised lattice structure under the uniform loading. The compliance of the lattice optimised roof is 36.8 kN · m, which means that the lattice volume can be reduced greatly by 35% with the compliance increased only by 3.5% compared with the shape optimised roof. It indicates that it is beneficial to remove some lattice struts without reducing the stiffness significantly.

The optimisation iterations are plotted in Figure 6.15. The shape optimisation converges after 48 iterations. The compliance jumps from 35.56 kN · m to 38.27 kN · m at the beginning of the lattice topology optimisation due to the reduction of lattice volume. The lattice topology optimisation converges after 20 iterations.

Chapter 7

Summary and future research

7.1 Summary

This dissertation proposes a new and systematic approach for the geometric design, structural analysis and optimisation of lattice-skin structures in the isogeometric framework. The performed research and the achievements of this thesis are as follows.

Lattice-skin geometry generation In order to generate lattice-skin structures in which the lattice is represented with a set of line segments and the thin-shell is represented with a subdivision surface, a new line/subdivision surface intersection algorithm is proposed in Chapter 3 to compute intersection points, including their physical coordinates as well as parametric coordinates.

The intersection algorithm consists of two sub-steps: intersection detection and intersection computation. A bounding volume tree of k-dops is constructed first for the subdivision surface in the intersection detection process, which aims to accelerate the process to detect the potential intersecting patches in the subdivision surface. The timing experiment with the Stanford bunny in Section 3.4 demonstrates clearly that k-dops are much more efficient than conventional bounding boxes for detecting potential intersections between the line and the subdivision surface when the lattice is not aligned with coordinate axes. After the subdivision patches that are possibly intersected with the line are detected, the patches are converted to corresponding Bézier representations.

The intersection computation between a line and a Bézier surface is performed with the implicit matrix representation of the Bézier surface, which has been described in detail in Section 3.2.3. Unlike the marching method and the divide-and-conquer method, the intersection computation with the implicit matrix representations is non-iterative.

In addition, the implicit matrix representations give the exact implicitisation of spline surfaces by evaluating the rank of the implicit matrix, which means that the accuracy of intersection points obtained is independent of the tolerance as used in the marching method or the divide-and-conquer method. The implicit matrix representations are also more general than traditional algebraic geometry methods for the implicitisation of spline surfaces. The Stanford bunny example given in Section 3.4 shows that the intersection computation with the implicit M-Reps achieves high accuracy with a moderate computational cost compared with the divide-and-conquer method.

Isogeometric analysis of lattice-skin structures Subdivision basis functions are used in the finite element analysis of the thin-shell such that the geometry model and the finite element model of the thin-shell are the same. The struts in the lattice-skin structure are modelled with truss elements. The lattice-skin coupling is considered with Lagrange multipliers, and the finite element discretisation of the equilibrium equations of the lattice-skin is also derived in Section 4.2. The parametric coordinates of coupled lattice nodes are obtained directly from the intersection computation with the implicit M-Rep. Therefore, the structural analysis of the lattice-skin structure is fully integrated with its geometry generation. The implementation of the finite element analysis of the lattice-skin structure is verified with the analysis of a sandwich plate, which shows that the numerical result agrees well with the analytical result. Lattice-skin structures with free-form surfaces are also generated and computed in the same framework as demonstrated in Section 4.3.

Lattice topology optimisation A new lattice topology optimisation method is proposed in Chapter 5 and it is further applied to the lattice-skin topology optimisation (see Section 6.1). The usual SIMP method used for continua is generalised to lattice structure applications. Several common benchmark examples from continuum topology optimisation are studied with the proposed topology optimisation method and the optimisation results are highly convincing as demonstrated in Chapter 5. The sensitivity analysis of the lattice-skin structure is derived in Section 6.1.1 for lattice topology optimisation which is based on lattice unit cells. The characteristic feature of each unit cell is a necessary information required in the method proposed, and it is preserved from the process of generating the lattice-skin geometry.

Shape optimisation with free-form deformation Shape optimisation using the free-form deformation technique is explored for thin-shells and lattice-skin structures in Chapter 6. The examples of shape optimisation of thin-shells demonstrate the feasibility of applying the free-form deformation method in shape optimisation. The shape sensitivity analysis of the lattice-skin structure is derived in Section 6.3.2, which involves the sensitivity analysis of the thin-shell and the lattice as well as the consideration of lattice-skin coupling. The shape optimisation algorithm of the lattice-skin structure proposed ensures that the lattice and the thin-shell remain coupled after updating both geometries during the optimisation process. Several examples of shape optimisation of lattice-skin structures are given in Section 6.4.

In summary, the specific contributions of this thesis are as follows:

- A novel isogeometric framework for integrated geometric design, analysis and optimisation is developed. The geometric design of the lattice and the skin are considered simultaneously so that all necessary information for structural analysis and optimisation, for example the parametric coordinates of the coupled lattice nodes on the thin-shell surface, is available throughout the design process. The integrated geometric design-analysis-optimisation framework ensures that there is no information loss in the whole design process and this process can be performed iteratively.

- A novel line/surface intersection algorithm combining a k-dop bounding volume tree and implicit matrix representations of spline surfaces is proposed [115], which enables to compute the intersection between a line and a spline surface in an one-shot manner without applying successive refinements or Newton-Raphson iterations. The efficiency and accuracy of the intersection computation are also demonstrated.

- A new lattice topology optimisation with SIMP-like regularisation and sensitivity filtering is proposed so that the SIMP method for topology optimisation of continua is generalised to lattice applications, and several benchmark examples from topology optimisation of continuum structures are used to demonstrate the feasibility of the proposed optimisation method.

- The free-form deformation technique has been applied in shape optimisation of lattice-skin structures, which enables the concurrent shape update of the lattice and the thin-shell while maintaining the structural coupling of the lattice and the skin.

7.2 Future research

Some possible improvements and extensions to the current research are listed as follows.

Lattice optimisation with constraints in additive manufacturing Though the additive manufacturing can fabricate objects with any complex geometries in theory, the quality of manufacturing can vary significantly depending on the geometry design, especially when overhanging surfaces exist in the geometry. The issue with overhanging surfaces has been reported in some research [116]. While topology optimisation of continuum structures considering constraints in additive manufacturing has been studied extensively [117–119], there are few reports on topology optimisation of lattice structures considering additive manufacturing constraints. A possible extension of the current research on lattice topology optimisation would be to incorporate shape optimisation of lattice structures to consider overhanging struts in the additive manufacturing.

Analysis of lattice-skin structures considering additive manufacturing process The fabrication factors in the additive manufacturing process can influence the mechanical properties of lattice materials, for example the residual thermal stress [120, 121]. To this end, the material model for the lattice structure may need some modifications to consider the influence. Alternatively, the manufacturing process can be analysed with the thermal effect considered.

Direct implicitisation of subdivision surfaces At present the subdivision patches are converted to corresponding Bézier patches, and the implicit matrix representations of Bézier patches are then obtained. However, for the subdivision patch containing extraordinary vertices, it cannot be represented exactly by a finite number of Bézier patches. It would be advantageous if an implicit matrix representation can be derived directly for a subdivision patch without converting to its Bézier representation. It would be interesting to investigate the possibility of deriving the implicit matrix representations near EVs.

Conformal lattice generation The conformal lattice structure is generated by moving internal lattice nodes to the nearest intersection points as described in Section 3.3. This process can be improved by projecting internal lattice nodes to the nearest point on the surface which is not necessarily the intersection point with the lattice line. This can be explored by computing the Euclidean distance between the point and the surface through the implicit matrix representation.

Lattice-skin optimisation with nonlinearity The assumption of small displacement is adopted in the present research for both structural analysis and optimisation. However, optimisation with nonlinearity is worth further exploration as the nonlinearity is common in practice, considering the large deformation and the nonlinear constitutive model of material. The issue of nonliearity can arise in the lattice-skin structure in cases where, for example, the buckling of the thin-shell between coupling lattice joints or the buckling of lattice struts in compressive stress. In order to consider the geometrical nonlinearity in the optimisation of lattice-skin structures, an iterative process should be included in order to obtain the equilibrium, as opposed to the linear case where the equilibrium corresponds to a linear system of equations. In addition, the sensitivity analysis is much more involved considering material nonlinearity since in general the tangent stiffness of the material is used in nonlinear analysis [122, 123].

Appendix A

Basis of box splines

A.1 Definition

Box splines represent a more general class of splines, which includes tensor-product B-splines as special cases and other splines that are not tensor-product, for example three-direction quartic box splines.

A box spline basis function can be defined with repeated convolution starting from a pulse function. For box spline curves, the pulse function defined in the direction vector $\boldsymbol{v}_1 = (1, 0)$ is

$$N_{\boldsymbol{v}_1}(\theta) = \begin{cases} 1, & \text{if } 0 < \theta < 1\,, \\ \frac{1}{2}, & \text{if } \theta = 0 \text{ or } \theta = 1\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{A.1}$$

For box spline surfaces, the pulse function is defined in terms of two direction vectors $\boldsymbol{v}_2 = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2\}$ given by, according to [75]

$$N_{\boldsymbol{v}_2}(\boldsymbol{\theta}) = \begin{cases} 1, & \text{if } \boldsymbol{\theta} = c_1\boldsymbol{\xi}_1 + c_2\boldsymbol{\xi}_2 \, (0 < c_1, c_2 < 1)\,; \\ \frac{1}{2}, & \text{if } \boldsymbol{\theta} = c_1\boldsymbol{\xi}_1 + c_2\boldsymbol{\xi}_2 \, (c_i = 0 \text{ or } 1 \text{ for exactly one index})\,; \\ \frac{1}{4}, & \text{if } \boldsymbol{\theta} = c_1\boldsymbol{\xi}_1 + c_2\boldsymbol{\xi}_2 \, (c_i = 0 \text{ or } 1 \text{ for both indices})\,; \\ 0, & \text{otherwise}\,, \end{cases} \tag{A.2}$$

where $\boldsymbol{\xi}_1 = (1, 0)$ and $\boldsymbol{\xi}_2 = (0, 1)$, c_1 and c_2 are coefficients in the two directions $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ respectively.

Box splines with higher orders can be generated by convolution of the pulse function with several additional direction vectors $\boldsymbol{\xi}_i$ $(i \geq 3)$ as follows,

$$N_{\boldsymbol{v}_i}(\boldsymbol{\theta}) = N_{\boldsymbol{v}_{i-1}\cup\boldsymbol{\xi}_i}(\boldsymbol{\theta}) = \int_0^1 N_{\boldsymbol{v}_{i-1}}(\boldsymbol{\theta} - t\boldsymbol{\xi}_i)\,\mathrm{d}t\,. \tag{A.3}$$

A.2 Generation of B-splines

B-splines are essentially special cases of box splines. For example, the linear, quadratic and cubic B-spline curves can be derived with the convolution in the following directions respectively,

$$\boldsymbol{v}_2 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{v}_3 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{v}_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{A.4}$$

A three-direction quartic box spline surface can be obtained by convolving along the directions [33]

$$\boldsymbol{v}_6 = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}, \tag{A.5}$$

and a bi-cubic B-spline surface can be derived by convolving along the directions

$$\boldsymbol{v}_8 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \tag{A.6}$$

A.3 Conversion to quartic Bézier basis

A regular Loop subdivision patch contains 12 control points, involving 12 corresponding quartic box spline basis functions [124], the transformation of these basis functions into the quartic Bézier basis arranged as in Figure 2.13 yields the following transformation matrix,

$$\boldsymbol{R} = \frac{1}{24}\begin{pmatrix}
0 & 0 & 0 & 2 & 2 & 0 & 2 & 12 & 2 & 0 & 2 & 2 \\
0 & 0 & 0 & 4 & 3 & 0 & 3 & 12 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 8 & 4 & 0 & 4 & 8 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 12 & 3 & 0 & 3 & 4 & 0 & 0 & 0 & 0 \\
2 & 2 & 2 & 12 & 2 & 0 & 2 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 1 & 0 & 4 & 12 & 0 & 0 & 3 & 1 \\
0 & 0 & 0 & 6 & 1 & 0 & 6 & 10 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 10 & 1 & 0 & 6 & 6 & 0 & 0 & 0 & 0 \\
1 & 0 & 3 & 12 & 1 & 0 & 4 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 8 & 8 & 0 & 0 & 4 & 0 \\
0 & 0 & 1 & 6 & 0 & 0 & 10 & 6 & 0 & 0 & 1 & 0 \\
0 & 0 & 4 & 8 & 0 & 0 & 8 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 3 & 0 & 0 & 12 & 4 & 0 & 1 & 3 & 0 \\
0 & 0 & 3 & 4 & 0 & 1 & 12 & 3 & 0 & 0 & 1 & 0 \\
0 & 0 & 2 & 2 & 0 & 2 & 12 & 2 & 0 & 2 & 2 & 0
\end{pmatrix}. \tag{A.7}$$

Appendix B

Thin-shell and sandwich panel formulations

B.1 Derivatives in thin-shell formulations

The covariant basis vectors (4.1) can be descretised as

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

where $\boldsymbol{x}_i$ are control points and $N_{i,\alpha}$ are first derivatives of associated shape functions.

The first derivatives of the covariant basis with respect to nodal coordinates are

$$\frac{\partial \boldsymbol{a}_\alpha}{\partial \boldsymbol{x}_i} = N_{i,\alpha} \mathbf{I} \,, \tag{B.2}$$

where $\mathbf{I}$ is an identity matrix since the first derivatives with respect to each component of nodal coordinates is required.

The first derivatives of the unit normal vector with respect to nodal coordinates are

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{x}_i} = \frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\boldsymbol{x}_i}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} - \frac{\boldsymbol{n} \cdot [\boldsymbol{n} \cdot (\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\boldsymbol{x}_i}]}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} \,, \tag{B.3}$$

considering that the following equation holds,

$$\begin{aligned} \frac{\partial |\boldsymbol{a}_1 \times \boldsymbol{a}_2|}{\partial \boldsymbol{x}_i} &= \frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2) \cdot (\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\boldsymbol{x}_i}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} \\ &= \boldsymbol{n} \cdot (\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\boldsymbol{x}_i} \,. \end{aligned} \tag{B.4}$$

Hence, the first derivatives of $\boldsymbol{n}_{,\alpha}$ with respect to nodal coordinates are

$$\begin{aligned}\frac{\partial \boldsymbol{n}_{,\alpha}}{\partial \boldsymbol{x}_i} &= \frac{\partial}{\partial \boldsymbol{x}_i}\left\{\frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} - \frac{\boldsymbol{n}\cdot[\boldsymbol{n}\cdot(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}]}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}\right\} \\ &= \frac{\partial}{\partial \boldsymbol{x}_i}\frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} - \boldsymbol{n}_{,\boldsymbol{x}_i}\cdot\frac{\boldsymbol{n}\cdot(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} \\ &\quad - \boldsymbol{n}\cdot\left[\boldsymbol{n}_{,\boldsymbol{x}_i}\cdot\frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}\right] - \boldsymbol{n}\cdot\left[\boldsymbol{n}\cdot\frac{\partial}{\partial \boldsymbol{x}_i}\frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}\right]\end{aligned} \tag{B.5}$$

with

$$\frac{\partial}{\partial \boldsymbol{x}_i}\frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} = \frac{[(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}]_{,\boldsymbol{x}_i}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|} - \frac{(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\alpha}[(\boldsymbol{a}_1 \times \boldsymbol{a}_2)_{,\boldsymbol{x}_i}]\cdot\boldsymbol{n}}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|^2}\,. \tag{B.6}$$

B.2 Displacement of isotropic sandwich panels

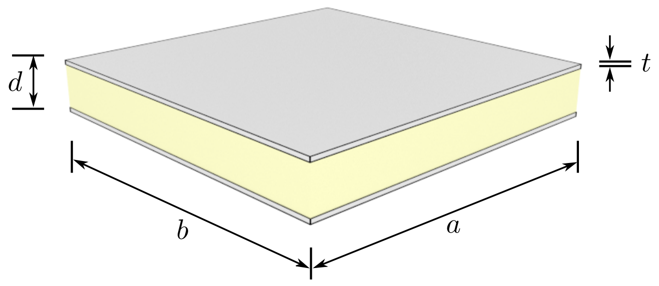


Fig. B.1 Sketch of a sandwich panel.

According to [103], the maximum displacement at the centre of the isotropic sandwich panel with identical thin-plate faces as shown in Figure B.1 is given by

$$w_{\max} = \frac{16qb^4}{\pi^6 D}\sum_m\sum_n\left[\frac{(-1)^{\frac{m-1}{2}}(-1)^{\frac{n-1}{2}}}{mn}\cdot\frac{1+\tau\varrho}{\varrho^2}\right] \tag{B.7}$$

with

$$D = \frac{Etd^2}{2(1-\nu^2)}\,, \tag{B.8}$$

$$\tau = \frac{\pi^2 Etd}{2(1-\nu^2)Gb^2}\,, \tag{B.9}$$

$$\varrho = \frac{m^2b^2}{a^2} + n^2\,, \tag{B.10}$$

where m and n are odd numbers $(1, 3, 5, \cdots)$; E, G and ν are the Young's modulus, the shear modulus, the Poisson's ratio of the material of the thin-plates; a and b are the dimensions of the thin-plates; t is the thickness of the thin-plates; d is the distance between the two thin-plates; q is the magnitude of the uniform loading. A detailed derivation of the displacement formula refers to [103].

Appendix C

Gradient-based optimisation algorithms

C.1 Sequential quadratic programming

The basic idea of sequential quadratic programming (SQP) is to approximate the nonlinear optimisation problem at each iteration with a quadratic programming subproblem and get a better approximation based on the solution to this subproblem. This process leads to a sequence of quadratic programming subproblems, and the iteration continues until some termination condition is satisfied.

Consider that $\{\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\}$ is the solution at the k-th iteration, and it is assumed that the solution in the next iteration $\{\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\phi}^{(k+1)}\}$ is closer to the local minimum $\{\boldsymbol{x}^{*}, \boldsymbol{\lambda}^{*}, \boldsymbol{\phi}^{*}\}$. The quadratic Taylor series approximation in $\boldsymbol{x}$ for the Lagrangian at the $(k+1)$-th iteration is

$$\begin{aligned}\mathcal{L}\Big(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big) \approx \mathcal{L}\Big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big) &+ \boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \mathcal{L}\Big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big) \\ &+ \frac{1}{2} \boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}} \nabla_{\boldsymbol{x}}^{2} \mathcal{L}\Big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big)\, \boldsymbol{\delta}_{\boldsymbol{x}}\,.\end{aligned} \tag{C.1}$$

It is reasonable to consider (C.1) as the objective function in the quadratic subproblem for the next iteration. Since $\mathcal{L}\Big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big)$ is a constant at the $(k+1)$-th iteration, the second and the third terms in the Taylor approximation (C.1) can then be considered as the objective function, i.e.

$$\boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}} \nabla_{\boldsymbol{x}} \mathcal{L}\Big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big) + \frac{1}{2} \boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}} \nabla_{\boldsymbol{x}}^{2} \mathcal{L}\Big(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\Big)\, \boldsymbol{\delta}_{\boldsymbol{x}}\,. \tag{C.2}$$

In practice, the Hessian of the Lagrangian $\nabla^2_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right)$ is approximated by the Hessian of the original objective function $\nabla^2_{\boldsymbol{x}}J\left(\boldsymbol{x}^{(k)}\right)$ in order to be able to solve the quadratic subproblem at any $\boldsymbol{x}^{(k)}$ and easier for a global convergence analysis [125]. However, the Hessian is not easy to compute in general as it involves the second derivatives of the objective and constraint functions. To cope with this issue, the exact Hessian is usually replaced with a quasi-Newton approximation [108]. In addition, $\nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right)$ is replaced by $\nabla_{\boldsymbol{x}}J\left(\boldsymbol{x}^{(k)}\right)$. As a matter of fact, this replacement leads to an equivalent subproblem when only equality constraints and their linearised approximations are considered. If inequality constraints exist, the replacement is not quite equivalent though it leads to an equivalent subproblems when the slack-variable formulation of the original optimisation problem is considered [125].

The linearised approximations of the constraints are considered with the first term of the Taylor series in $\boldsymbol{x}$

$$G_i\left(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) \approx G_i\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) + \delta_{\boldsymbol{x}}^{\mathsf{T}}\nabla_{\boldsymbol{x}}G_i\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) , \tag{C.3}$$

$$H_j\left(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) \approx H_j\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) + \delta_{\boldsymbol{x}}^{\mathsf{T}}\nabla_{\boldsymbol{x}}H_j\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) . \tag{C.4}$$

Therefore, the quadratic subproblem at the $(k+1)$-th iteration is of the form

$$\begin{aligned} &\text{minimise} && \boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}}\nabla_{\boldsymbol{x}}J\left(\boldsymbol{x}^{(k)}\right) + \frac{1}{2}\boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}}\nabla^2_{\boldsymbol{x}}J\left(\boldsymbol{x}^{(k)}\right)\boldsymbol{\delta}_{\boldsymbol{x}} && \text{(C.5a)} \\ &\text{subject to} && G_i\left(\boldsymbol{x}^{(k)}\right) + \delta_{\boldsymbol{x}}^{\mathsf{T}}\nabla_{\boldsymbol{x}}G_i\left(\boldsymbol{x}^{(k)}\right) = 0, \quad i = 1, \cdots, n_p && \text{(C.5b)} \\ &&& H_j\left(\boldsymbol{x}^{(k)}\right) + \delta_{\boldsymbol{x}}^{\mathsf{T}}\nabla_{\boldsymbol{x}}H_j\left(\boldsymbol{x}^{(k)}\right) \leq 0, \quad j = 1, \cdots, n_q && \text{(C.5c)} \end{aligned}$$

The solution $\boldsymbol{\delta}_{\boldsymbol{x}}^*$ of the quadratic subproblem (C.5) can be used to calculate the next iterate $\boldsymbol{x}^{(k+1)}$ by updating $\boldsymbol{x}^{(k)}$ in the direction of $\boldsymbol{\delta}_{\boldsymbol{x}}^*$. For the new iterates of multipliers, one possible set of candidates are the corresponding optimal multipliers of (C.5). The validity of using the optimal multipliers of the quadratic subproblem can be seen from the following analysis.

Considering the linear approximation of the KKT condition (5.3a) yields,

$$\begin{aligned} \nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\phi}^{(k+1)}\right) \approx & \nabla_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) + \boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}}\nabla^2_{\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) \\ & + \boldsymbol{\delta}_{\boldsymbol{\lambda}}^{\mathsf{T}}\nabla^2_{\boldsymbol{\lambda}\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) + \boldsymbol{\delta}_{\boldsymbol{\phi}}^{\mathsf{T}}\nabla^2_{\boldsymbol{\phi}\boldsymbol{x}}\mathcal{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\phi}^{(k)}\right) \end{aligned} \tag{C.6}$$

with

$$\nabla_{\boldsymbol{x}}\mathcal{L}\Big(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\phi}^{(k)}\Big)=\nabla_{\boldsymbol{x}}J\Big(\boldsymbol{x}^{(k)}\Big)+\sum_{i=1}^{n_p}\lambda_i^{(k)}\nabla_{\boldsymbol{x}}G_i\Big(\boldsymbol{x}^{(k)}\Big)+\sum_{j=1}^{n_q}\phi_j^{(k)}\nabla_{\boldsymbol{x}}H_j\Big(\boldsymbol{x}^{(k)}\Big)\ , \tag{C.7}$$

$$\nabla^2_{\boldsymbol{x}}\mathcal{L}\Big(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\phi}^{(k)}\Big)=\nabla^2_{\boldsymbol{x}}J\Big(\boldsymbol{x}^{(k)}\Big)+\sum_{i=1}^{n_p}\lambda_i^{(k)}\nabla^2_{\boldsymbol{x}}G_i\Big(\boldsymbol{x}^{(k)}\Big)+\sum_{j=1}^{n_q}\phi_j^{(k)}\nabla^2_{\boldsymbol{x}}H_j\Big(\boldsymbol{x}^{(k)}\Big)\ , \tag{C.8}$$

$$\nabla^2_{\boldsymbol{\lambda x}}\mathcal{L}\Big(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\phi}^{(k)}\Big)=\nabla_{\boldsymbol{x}}\boldsymbol{G}\Big(\boldsymbol{x}^{(k)}\Big)=\Big(\nabla_{\boldsymbol{x}}G_1(\boldsymbol{x}^{(k)})\,\nabla_{\boldsymbol{x}}G_2(\boldsymbol{x}^{(k)})\ \cdots\ \nabla_{\boldsymbol{x}}G_{n_p}(\boldsymbol{x}^{(k)})\Big)^{\mathsf{T}}\ , \tag{C.9}$$

$$\nabla^2_{\boldsymbol{\phi x}}\mathcal{L}\Big(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\phi}^{(k)}\Big)=\nabla_{\boldsymbol{x}}\boldsymbol{H}\Big(\boldsymbol{x}^{(k)}\Big)=\Big(\nabla_{\boldsymbol{x}}H_1(\boldsymbol{x}^{(k)})\,\nabla_{\boldsymbol{x}}H_2(\boldsymbol{x}^{(k)})\ \cdots\ \nabla_{\boldsymbol{x}}H_{n_q}(\boldsymbol{x}^{(k)})\Big)^{\mathsf{T}}\ . \tag{C.10}$$

Substituting (C.7), (C.8), (C.9) and (C.10) into (C.6) yields

$$\nabla_{\boldsymbol{x}}J\Big(\boldsymbol{x}^{(k)}\Big)+\boldsymbol{\delta}_{\boldsymbol{x}}^{\mathsf{T}}\nabla^2_{\boldsymbol{x}}\mathcal{L}\Big(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\phi}^{(k)}\Big)+\boldsymbol{\lambda}_{k+1}^{\mathsf{T}}\nabla_{\boldsymbol{x}}\boldsymbol{G}\Big(\boldsymbol{x}^{(k)}\Big)+\boldsymbol{\phi}_{k+1}^{\mathsf{T}}\nabla_{\boldsymbol{x}}\boldsymbol{H}\Big(\boldsymbol{x}^{(k)}\Big)=0\,. \tag{C.11}$$

Given $\Big(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\phi}^{(k)}\Big)$, (C.11) is exactly the first derivative condition of the quadratic subproblem (C.5) at a local minimum $\boldsymbol{\delta}_{\boldsymbol{x}}^*$ when the Hessian of the Lagrangian is approximated with the Hessian of the original objective function. Therefore, the optimal multipliers $\{\boldsymbol{\lambda}_{k+1}^*,\boldsymbol{\phi}_{k+1}^*\}$ obtained from (C.5) is an appropriate choice for the next iterates of multipliers $\boldsymbol{\lambda}^{(k+1)}$ and $\boldsymbol{\phi}^{(k+1)}$. A modification of the next iterates $\{\boldsymbol{x}^{(k+1)},\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\phi}^{(k+1)}\}$ would be a line search along the directions $\{\boldsymbol{\delta}_{\boldsymbol{x}},\boldsymbol{\delta}_{\boldsymbol{\lambda}},\boldsymbol{\delta}_{\boldsymbol{\phi}}\}$ where

$$\boldsymbol{\delta}_{\boldsymbol{x}}=\boldsymbol{\delta}_{\boldsymbol{x}}^*,\quad \boldsymbol{\delta}_{\boldsymbol{\lambda}}=\boldsymbol{\lambda}_{k+1}^*-\boldsymbol{\lambda}^{(k)},\quad \boldsymbol{\delta}_{\boldsymbol{\phi}}=\boldsymbol{\phi}_{k+1}^*-\boldsymbol{\phi}^{(k)}\,. \tag{C.12}$$

Therefore, the next iterates at $(k+1)$-th iteration are

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\varepsilon^{(k)}\boldsymbol{\delta}_{\boldsymbol{x}},\quad \boldsymbol{\lambda}^{(k+1)}=\boldsymbol{\lambda}^{(k)}+\varepsilon^{(k)}\boldsymbol{\delta}_{\boldsymbol{\lambda}},\quad \boldsymbol{\phi}^{(k+1)}=\boldsymbol{\phi}^{(k)}+\varepsilon^{(k)}\boldsymbol{\delta}_{\boldsymbol{\phi}}\,, \tag{C.13}$$

where $\varepsilon^{(k)}$ is the step length which can be obtained by a line search.

One advantage of using SQP to solve nonlinear optimisation problems is that neither the initial point $\boldsymbol{x}_0$ nor the iterates $\boldsymbol{x}^{(k)}$ need to be feasible. In addition, SQP decomposes the nonlinear problem into a sequence of quadratic programming subproblems with linear constraints, which are relatively easier to solve as there are some good algorithms for the quadratic programming problems [108].

C.2 Method of moving asymptotes

The method of moving asymptotes (MMA) was first proposed by Svanberg [126]. It can be seen as a generalisation of the convex linearisation method developed by Fleury et al. [127]. In the MMA, each function at every iteration is linearised in terms of the intervening variables $1/\left(x_j - x_j^L\right)$ or $1/\left(x_j^U - x_j\right)$, where x_j^L and x_j^U are the moving asymptotes that can be used to control the conservation and stability of the optimisation process. The linearisation of functions in the MMA is as follows.

Given an iterate $\boldsymbol{x}^{(k)}$ at the k-th iteration, the values of moving asymptotes $x_j^{L(k)}$ and $x_j^{U(k)}$ are chosen such that

$$x_j^{L(k)} < x_j^{(k)} < x_j^{U(k)}, \quad j = 1, \cdots, n\,. \tag{C.14}$$

The linear approximation $f^{(k)}$ of a function f at the k-th iteration can be defined as

$$\begin{aligned} f^{(k)}(\boldsymbol{x}) &= f\left(\boldsymbol{x}^{(k)}\right) - \sum_{j=1}^{n} \left(\frac{f_j^{U(k)}}{x_j^{U(k)} - x_j^{(k)}} + \frac{f_j^{L(k)}}{x_j^{(k)} - x_j^{L(k)}} \right) + \sum_{j=1}^{n} \left(\frac{f_j^{U(k)}}{x_j^{U(k)} - x_j} + \frac{f_j^{L(k)}}{x_j - f_j^{L(k)}} \right) \\ &= f\left(\boldsymbol{x}^{(k)}\right) + \sum_{j=1}^{n} f_j^{U(k)} \left(\frac{1}{x_j^{U(k)} - x_j} - \frac{1}{x_j^{U(k)} - x_j^{(k)}} \right) \\ &\qquad + \sum_{j=1}^{n} f_j^{L(k)} \left(\frac{1}{x_j - x_j^{L(k)}} - \frac{1}{x_j^{(k)} - x_j^{L(k)}} \right), \end{aligned} \tag{C.15}$$

where f can be either the objective function or constraint functions in the optimisation problem, and

$$f_j^{U(k)} = \begin{cases} \left(x_j^{U(k)} - x_j^{(k)}\right)^2 \dfrac{\partial f}{\partial x_j}, & \text{if } \dfrac{\partial f}{\partial x_j} > 0\,; \\ \\ 0, & \text{otherwise}\,; \end{cases} \tag{C.16}$$

$$f_j^{L(k)} = \begin{cases} -\left(x_j^{(k)} - x_j^{L(k)}\right)^2 \dfrac{\partial f}{\partial x_j}, & \text{if } \dfrac{\partial f}{\partial x_j} < 0\,; \\ \\ 0, & \text{otherwise}\,. \end{cases} \tag{C.17}$$

After substituting (C.16) and (C.17) into (C.15), the linear approximation in MMA becomes

$$f^{(k)}(\boldsymbol{x}) = \begin{cases} f\left(\boldsymbol{x}^{(k)}\right) + \sum\limits_{j=1}^{n} \dfrac{f_j^{U(k)} - x_j^{(k)}}{f_j^{U(k)} - x_j} \dfrac{\partial f}{\partial x_j} \left(x_j - x_j^{(k)}\right), & \text{if } \dfrac{\partial f}{\partial x_j} > 0 \\ f\left(\boldsymbol{x}^{(k)}\right) + \sum\limits_{j=1}^{n} \dfrac{x_j^{(k)} - f_j^{L(k)}}{x_j - f_j^{L(k)}} \dfrac{\partial f}{\partial x_j} \left(x_j - x_j^{(k)}\right), & \text{if } \dfrac{\partial f}{\partial x_j} < 0 \\ f\left(\boldsymbol{x}^{(k)}\right), & \text{if } \dfrac{\partial f}{\partial x_j} = 0 \end{cases} \tag{C.18}$$

It can be observed from (C.18) that $f^{(k)}$ is a linear approximation of f at $\boldsymbol{x}^{(k)}$. Specifically, $f^{(k)}\left(\boldsymbol{x}^{(k)}\right) = f\left(\boldsymbol{x}^{(k)}\right)$ and $\partial f^{(k)}/\partial x_j = \partial f/\partial x_j$ at $\boldsymbol{x} = \boldsymbol{x}^{(k)}$. The second derivatives of the approximation function $f^{(k)}$ are

$$\frac{\partial^2 f^{(k)}}{\partial x_j^2} = \frac{2f_j^{U(k)}}{\left(x_j^{U(k)} - x_j\right)^3} + \frac{2f_j^{L(k)}}{\left(x_j - x_j^{L(k)}\right)^3}, \tag{C.19}$$

$$\frac{\partial^2 f^{(k)}}{\partial x_j \partial x_k} = 0, \quad \text{if } j \neq k\,. \tag{C.20}$$

As can be seen, the second derivatives of $f^{(k)}$ are always nonnegative, which indicates that it is a convex function. Furthermore, the second derivatives at $\boldsymbol{x}^{(k)}$ become larger when the moving asymptotes $x_j^{L(k)}$ and $x_j^{U(k)}$ are closer to $\boldsymbol{x}^{(k)}$, indicating a larger curvature of the approximating function $f^{(k)}$ in the neighbourhood of $\boldsymbol{x}^{(k)}$. On the other hand, if $x_j^{L(k)}$ and $x_j^{U(k)}$ move far away from $\boldsymbol{x}^{(k)}$, $f^{(k)}$ becomes more linear around $\boldsymbol{x}^{(k)}$. For example in the extreme case where $x_j^{L(k)} = -\infty$ and $x_j^{U(k)} = \infty$, the approximating function $f^{(k)}$ becomes a linear function that is identical to the one adopted in the sequential linear programming.

The effect of the moving asymptotes $x_j^{L(k)}$ and $x_j^{U(k)}$ on the optimisation process is examined as follows. Consider two sets of moving asymptotes $\{x_j^{L(k)}, x_j^{U(k)}\}$ and $\{\widetilde{x}_j^{L(k)}, \widetilde{x}_j^{U(k)}\}$ satisfying

$$x_j^{L(k)} \leq \widetilde{x}_j^{L(k)} < x_j^{(k)} < \widetilde{x}_j^{U(k)} \leq x_j^{U(k)}, \tag{C.21}$$

the difference of the approximating functions $f^{(k)}$ and $\widetilde{f}^{(k)}$ defined in terms of $\{x_j^{L(k)}, x_j^{U(k)}\}$ and $\{\widetilde{x}_j^{L(k)}, \widetilde{x}_j^{U(k)}\}$ respectively is

$$\Delta f^{(k)} = \widetilde{f}^{(k)} - f^{(k)}\,. \tag{C.22}$$

It can be shown that $\Delta f^{(k)}(x_j^{(k)}) = 0$ and $\partial \Delta f^{(k)}/\partial x_j = 0$ at $x_j = x_j^{(k)}$. Recall that $\partial f^{(k)}/\partial x_j = \partial f/\partial x_j$ at $\boldsymbol{x} = \boldsymbol{x}^{(k)}$. Hence, $\partial f^{(k)}/\partial x_j = \partial \widetilde{f}^{(k)}/\partial x_j$ at $\boldsymbol{x} = \boldsymbol{x}^{(k)}$. The second derivative of $\Delta f^{(k)}$ with respect to x_j is

$$\left.\frac{\partial^2 \Delta f^{(k)}}{\partial x_j^2}\right|_{x_j = x_j^{(k)}} = \begin{cases} \dfrac{2}{\widetilde{x}_j^{U(k)} - x_j^{(k)}} \dfrac{\partial f}{\partial x_j} - \dfrac{2}{x_j^{U(k)} - x_j^{(k)}} \dfrac{\partial f}{\partial x_j}, & \text{if } \dfrac{\partial f}{\partial x_j} \geq 0 \\ -\dfrac{2}{x_j^{(k)} - \widetilde{x}_j^{L(k)}} \dfrac{\partial f}{\partial x_j} + \dfrac{2}{x_j^{(k)} - x_j^{L(k)}} \dfrac{\partial f}{\partial x_j}, & \text{if } \dfrac{\partial f}{\partial x_j} < 0 \end{cases}. \tag{C.23}$$

Since $\partial^2 \Delta f^{(k)}/\partial x_j^2$ is always nonnegative at $x_j = x_j^{(k)}$, $\boldsymbol{x}^{(k)}$ is a local minimum of $\Delta f^{(k)}$. Hence, $\Delta f^{(k)} \geq 0$, i.e. the function value approximated with $f^{(k)}$ is larger than the one approximated with $\widetilde{f}^{(k)}$. Therefore, using moving asymptotes closer to the iterate $\boldsymbol{x}^{(k)}$ leads to a more conservative approximation of the original function. This can be used to control the conservation and stability during the optimisation process by adjusting the values of moving asymptotes. If the optimisation process appears some oscillation, some closer moving asymptotes can be chosen; in the case of a slow convergence of the optimisation process, the asymptotes can be moved away from the current iterate.

The effect of moving asymptotes is illustrated in Figure C.1 plotting the function $f = x^2$ and its approximations f^M at $x = 1$ in the MMA considering different moving asymptote $x^U = 1.2, 1.5, 2$ and 10. It is shown that with a larger value of x^U, the approximation is more linear; on the other hand, the curvature of the approximation function is larger with a smaller value of x^U.

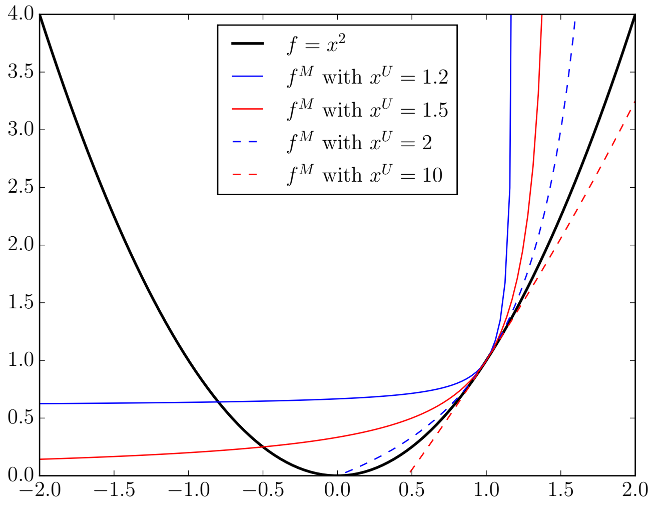


Fig. C.1 Approximating functions in MMA with different upper moving asymptote values.

Appendix D

Stiffness transformation matrix and its derivatives

In the case of three dimensional lattice structures modelled with beam elements, the transformation matrix used to compute the element stiffness matrix takes the following form

$$\boldsymbol{\Lambda}_j = diag\{\boldsymbol{\Lambda}_j^0, \boldsymbol{\Lambda}_j^0, \boldsymbol{\Lambda}_j^0, \boldsymbol{\Lambda}_j^0\} \tag{D.1}$$

with

$$\boldsymbol{\Lambda}_j^0 = \begin{pmatrix} \cos\alpha^{\mathrm{l}}\cos\beta^{\mathrm{l}} & \sin\alpha^{\mathrm{l}}\cos\beta^{\mathrm{l}} & -\sin\beta^{\mathrm{l}} \\ -\sin\alpha^{\mathrm{l}} & \cos\alpha^{\mathrm{l}} & 0 \\ \cos\alpha^{\mathrm{l}}\sin\beta^{\mathrm{l}} & \sin\alpha^{\mathrm{l}}\sin\beta^{\mathrm{l}} & \cos\beta^{\mathrm{l}} \end{pmatrix}, \tag{D.2}$$

where j is the index of the element, α^{l} is the angle between the projection of the element on XY plane and X axis, and β^{l} is the angle between the element and Z axis. In the following context, for the sake of symbol simplicity and without loss of generality, we drop the symbol j and all variables are derived for one element.

A simple derivation is as follows. Suppose a local coordinate system is attached to each strut with the local x-axis aligned with the strut direction. The local coordinates $\overline{\boldsymbol{x}}^{\mathrm{l}}$ can be obtained by mapping coordinates $\boldsymbol{x}^{\mathrm{l}}$ in a global coordinate system with a transformation matrix

$$\boldsymbol{\Lambda} = \left(\mathbf{R}_z^{\mathrm{l}}(\alpha^{\mathrm{l}})\mathbf{R}_y^{\mathrm{l}}(\beta^{\mathrm{l}})\right)^{\mathsf{T}}, \tag{D.3}$$

that is, $\overline{\boldsymbol{x}}^{\mathrm{l}} = \boldsymbol{\lambda}\boldsymbol{x}^{\mathrm{l}}$, where $\mathbf{R}_z^{\mathrm{l}}$ and $\mathbf{R}_y^{\mathrm{l}}$ are rotation matrices about z- and y-axes, respectively; α and β are Euler angles describing the orientation of the strut in the global coordinate system as in (D.2).

With the transformation matrix, the lattice element stiffness matrix can be computed with

$$\mathbf{k}^e = \mathbf{\Lambda}^{\mathsf{T}} \overline{\mathbf{k}}^e \mathbf{\Lambda} \tag{D.4}$$

where $\overline{\mathbf{k}}^e$ is the lattice element stiffness matrix in the local coordinate system which is straightforward to obtain.

Equation (D.2) can be expressed in terms of nodal coordinates of an element as

$$\mathbf{\Lambda}^0 = \begin{pmatrix} C_x & C_y & C_z \\ -C_y^{\mathrm{p}} & C_x^{\mathrm{p}} & 0 \\ -C_x^{\mathrm{p}} C_z & -C_y^{\mathrm{p}} C_z & L^{\mathrm{p}}/L \end{pmatrix} \tag{D.5}$$

with

$$C_x = \frac{x_2 - x_1}{L}, \quad C_y = \frac{y_2 - y_1}{L}, \quad C_z = \frac{z_2 - z_1}{L}\,, \tag{D.6}$$

$$C_x^{\mathrm{p}} = \frac{x_2 - x_1}{L^{\mathrm{p}}}, \quad C_y^{\mathrm{p}} = \frac{y_2 - y_1}{L^{\mathrm{p}}}, \quad C_z^{\mathrm{p}} = \frac{z_2 - z_1}{L^{\mathrm{p}}}\,, \tag{D.7}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) are two end nodes of the element, L is the element length, L^{p} the the length projected on XY plane.

Furthermore, equation (D.5) can be written in a form as

$$\mathbf{\Lambda}^0 = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \boldsymbol{c}_3 \end{pmatrix} \tag{D.8}$$

with

$$\boldsymbol{c}_1 = \begin{pmatrix} C_x & C_y & C_z \end{pmatrix}\,, \tag{D.9}$$

$$\boldsymbol{c}_2 = \begin{pmatrix} -C_y^{\mathrm{p}} & C_x^{\mathrm{p}} & 0 \end{pmatrix}\,, \tag{D.10}$$

$$\boldsymbol{c}_3 = \begin{pmatrix} -C_x^{\mathrm{p}} C_z & -C_y^{\mathrm{p}} C_z & L^{\mathrm{p}}/L \end{pmatrix}\,. \tag{D.11}$$

The first derivatives of $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are

$$\frac{\partial \boldsymbol{c}_1}{\partial x_p^q} = \frac{1}{L} \boldsymbol{e}_1^{\mathsf{T}} (\mathbf{I} - \boldsymbol{c}_1^{\mathsf{T}} \boldsymbol{c}_1)\,, \tag{D.12}$$

$$\frac{\partial \boldsymbol{c}_2}{\partial x_p^q} = \frac{1}{L^{\mathrm{p}}} \boldsymbol{e}_2^{\mathsf{T}} (\mathbf{I} - \boldsymbol{c}_2^{\mathsf{T}} \boldsymbol{c}_2) \tag{D.13}$$

with

$$\boldsymbol{e}_1^{\mathsf{T}} = \begin{cases} (-1 \quad 0 \quad 0) & \text{for } x_1^1 \\ (0 \quad -1 \quad 0) & \text{for } x_1^2 \\ (0 \quad 0 \quad -1) & \text{for } x_1^3 \\ (1 \quad 0 \quad 0) & \text{for } x_2^1 \\ (0 \quad 1 \quad 0) & \text{for } x_2^2 \\ (0 \quad 0 \quad 1) & \text{for } x_2^3 \end{cases} \quad \text{and} \quad \boldsymbol{e}_2^{\mathsf{T}} = \begin{cases} (0 \quad -1 \quad 0) & \text{for } x_1^1 \\ (1 \quad 0 \quad 0) & \text{for } x_1^2 \\ (0 \quad 0 \quad 0) & \text{for } x_1^3 \\ (0 \quad 1 \quad 0) & \text{for } x_2^1 \\ (-1 \quad 0 \quad 0) & \text{for } x_2^2 \\ (0 \quad 0 \quad 0) & \text{for } x_2^3 \end{cases} . \tag{D.14}$$

Before giving the expression for the first derivative of $\boldsymbol{c}_3$, $\boldsymbol{c}_3$ is written as

$$\begin{aligned} \boldsymbol{c}_3 &= -\left(C_x^{\mathrm{p}} \quad C_y^{\mathrm{p}} \quad 0\right) C_z + (0 \quad 0 \quad L^{\mathrm{p}}/L) \\ &= -\boldsymbol{c}_3^1 C_z + \boldsymbol{c}_3^2 . \end{aligned} \tag{D.15}$$

Hence,

$$\frac{\partial \boldsymbol{c}_3}{\partial x_p^q} = -\frac{\partial \boldsymbol{c}_3^1}{\partial x_p^q} C_z - \boldsymbol{c}_3^1 \frac{\partial C_z}{\partial x_p^q} + \frac{\partial \boldsymbol{c}_3^2}{\partial x_p^q} \tag{D.16}$$

with

$$\frac{\partial \boldsymbol{c}_3^1}{\partial x_p^q} = \frac{1}{L^{\mathrm{p}}} \boldsymbol{e}_3^{\mathsf{T}} (\mathbf{I} - \boldsymbol{c}_3^{1\mathsf{T}} \boldsymbol{c}_3^1) , \tag{D.17}$$

$$\frac{\partial \boldsymbol{c}_3^2}{\partial x_p^q} = \frac{1}{L^2} (L \boldsymbol{c}_3^1 \boldsymbol{e}_3 - L^{\mathrm{p}} \boldsymbol{c}_1 \boldsymbol{e}_1) , \tag{D.18}$$

where

$$\boldsymbol{e}_3^{\mathsf{T}} = \begin{cases} (-1 \quad 0 \quad 0) & \text{for } x_1^1 \\ (0 \quad -1 \quad 0) & \text{for } x_1^2 \\ (0 \quad 0 \quad 0) & \text{for } x_1^3 \\ (1 \quad 0 \quad 0) & \text{for } x_2^1 \\ (0 \quad 1 \quad 0) & \text{for } x_2^2 \\ (0 \quad 0 \quad 0) & \text{for } x_2^3 \end{cases} , \tag{D.19}$$

and $\partial C_z / \partial x_p^q$ can be obtained directly from the third column of $\partial \boldsymbol{c}_1 / \partial x_p^q$.

References

[1] R. Lakes. Materials with structural hierarchy. *Nature*, 361:511, 1993.

[2] J. Aizenberg, J. C. Weaver, M. S. Thanawala, V. C. Sundar, D. E. Morse, and P. Fratzl. Skeleton of euplectella sp.: structural hierarchy from the nanoscale to the macroscale. *Science*, 309:275–278, 2005.

[3] L. J. Gibson and M. F. Ashby. *Cellular solids: structure and properties*. Cambridge university press, 1999.

[4] L. J. Gibson, M. F. Ashby, and B. A. Harley. *Cellular materials in nature and medicine*. Cambridge University Press, 2010.

[5] W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C. B. Williams, C. C.L. Wang, Y. C. Shin, S. Zhang, and P. D. Zavattieri. The status, challenges, and future of additive manufacturing in engineering. *Computer-Aided Design*, 69:65–89, 2015.

[6] U. G.K. Wegst, H. Bai, E. Saiz, A. P. Tomsia, and R. O. Ritchie. Bioinspired structural materials. *Nature Materials*, 14:23, 2015.

[7] T. A. Schaedler, A. J. Jacobsen, A. Torrents, A. E. Sorensen, J. Lian, J. R. Greer, L. Valdevit, and W. B. Carter. Ultralight metallic microlattices. *Science*, 334:962–965, 2011.

[8] X. Zheng, H. Lee, T. H. Weisgraber, M. Shusteff, J. DeOtte, E. B. Duoss, J. D. Kuntz, M. M. Biener, Q. Ge, J. A. Jackson, et al. Ultralight, ultrastiff mechanical metamaterials. *Science*, 344:1373–1377, 2014.

[9] L. R. Meza, S. Das, and J. R. Greer. Strong, lightweight, and recoverable three-dimensional ceramic nanolattices. *Science*, 345:1322–1326, 2014.

[10] ASTM f2792. Standard terminology for additive manufacturing technologies. Standard, ASTM International, 2012.

[11] 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*, 194:4135–4195, 2005.

[12] P. Kagan, A. Fischer, and P. Z. Bar-Yoseph. New B-spline finite element approach for geometrical design and mechanical analysis. *International Journal for Numerical Methods in Engineering*, 41:435–458, 1998.

[13] 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*, 47:2039–2072, 2000.

[14] F. Cirak, M. J. Scott, E. K. Antonsson, M. Ortiz, and P. Schröder. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *Computer-Aided Design*, 34:137–148, 2002.

[15] F. Cirak and Q. Long. Subdivision shells with exact boundary control and non-manifold geometry. *International Journal for Numerical Methods in Engineering*, 88:897–923, 2011.

[16] Q. Long, P. Burkhard Bornemann, and F. Cirak. Shear-flexible subdivision shells. *International Journal for Numerical Methods in Engineering*, 90:1549–1577, 2012.

[17] Q. Zhang, M. Sabin, and F. Cirak. Subdivision surfaces with isogeometric analysis adapted refinement weights. *Computer-Aided Design*, 102:104–114, 2018.

[18] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 197:2976–2988, 2008.

[19] K. Bandara, T. Rüberg, and F. Cirak. Shape optimisation with multiresolution subdivision surfaces and immersed finite elements. *Computer Methods in Applied Mechanics and Engineering*, 300:510–539, 2016.

[20] K. Bandara and F. Cirak. Isogeometric shape optimisation of shell structures using multiresolution subdivision surfaces. *Computer-Aided Design*, 95:62–71, 2018.

[21] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J.R. Hughes. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics*, 38:310–322, 2006.

[22] Y. Bazilevs, V. M. Calo, T. J.R. Hughes, and Y. Zhang. Isogeometric fluid-structure interaction: theory, algorithms, and computations. *Computational Mechanics*, 43:3–37, 2008.

[23] T. Rüberg and F. Cirak. A fixed-grid b-spline finite element technique for fluid–structure interaction. *International Journal for Numerical Methods in Fluids*, 74:623–660, 2014.

[24] A. J. Cottrell, T. J.R. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[25] 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*, 199:229–263, 2010.

[26] M. R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199:264–275, 2010.

[27] M. A. Scott, R. N. Simpson, J.A. Evans, S. Lipton, S. P.A. Bordas, T. J.R. Hughes, and T. W. Sederberg. Isogeometric boundary element analysis using unstructured T-splines. *Computer Methods in Applied Mechanics and Engineering*, 254:197–221, 2013.

[28] F. Cirak and M. Ortiz. Fully c1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 51:813–833, 2001.

[29] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Transactions on Graphics*, 22:477–484, 2003.

[30] X. Li, J. Zheng, T. W. Sederberg, T. J.R. Hughes, and M. A. Scott. On linear independence of T-spline blending functions. *Computer Aided Geometric Design*, 29:63–76, 2012.

[31] M. A. Scott, X. Li, T. W. Sederberg, and T. J.R. Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213:206–222, 2012.

[32] 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*, 199:1437–1445, 2010.

[33] L. Andersson and N. F. Stewart. *Introduction to the mathematics of subdivision surfaces.* Siam, 2010.

[34] T. J. Cashman, U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. *ACM Transactions on Graphics*, 28:46, 2009.

[35] A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner. Isogeometric shell analysis with NURBS compatible subdivision surfaces. *Applied Mathematics and Computation*, 272:139–147, 2016.

[36] I. Gibson, D. W. Rosen, and B. Stucker. *Additive manufacturing technologies.* Springer, 2010.

[37] M. K. Thompson, G. Moroni, T. Vaneker, G. Fadel, R. I. Campbell, I. Gibson, A. Bernard, J. Schulz, P. Graf, B. Ahuja, et al. Design for additive manufacturing: Trends, opportunities, considerations, and constraints. *CIRP Annals*, 65:737–760, 2016.

[38] H. Wang, Y. Chen, and D. W. Rosen. A hybrid geometric modeling method for large scale conformal cellular structures. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 421–427. American Society of Mechanical Engineers, 2005.

[39] C. Chu, G. Graf, and D. W. Rosen. Design for additive manufacturing of cellular structures. *Computer-Aided Design and Applications*, 5:686–696, 2008.

[40] L. A Piegl and A. M. Richard. Tessellating trimmed nurbs surfaces. *Computer-Aided Design*, 27:16–26, 1995.

[41] L. A Piegl and W. Tiller. Geometry-based triangulation of trimmed nurbs surfaces. *Computer-Aided Design*, 30:11–18, 1998.

[42] C. L. Bajaj, C. M. Hoffmann, R. E. Lynch, and J.E.H. Hopcroft. Tracing surface intersections. *Computer aided geometric design*, 5:285–307, 1988.

[43] R. E. Barnhill and S.N. Kersey. A marching method for parametric surface/surface intersection. *Computer aided geometric design*, 7:257–280, 1990.

[44] T. W. Sederberg and J. Zheng. *Algebraic methods for computer aided geometric design.* North-Holland, Amsterdam, 2002.

[45] J. Cox, D.and Little and D. O'shea. *Ideals, varieties, and algorithms*, volume 3. Springer, 2007.

[46] D. A. Cox, J. Little, and D. O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[47] T. W. Sederberg, T. Saito, D. Qi, and K. S. Klimaszewski. Curve implicitization using moving lines. *Computer Aided Geometric Design*, 11:687–706, 1994.

[48] T. W Sederberg and F. Chen. Implicitization using moving curves and surfaces. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 301–308. ACM, 1995.

[49] D. M. Y. Sommerville. *Analytical geometry of three dimensions.* Cambridge University Press, 1959.

[50] L. Busé and T. L. Ba. Matrix-based implicit representations of rational algebraic curves and applications. *Computer Aided Geometric Design*, 27:681–699, 2010.

[51] L. Busé. Implicit matrix representations of rational Bézier curves and surfaces. *Computer-Aided Design*, 46:14–24, 2014.

[52] M. P. Bendsøe and O. Sigmund. *Topology optimization: theory, method and applications.* Springer, 2003.

[53] G. I.N. Rozvany and T. Lewiński. *Topology optimization in structural and continuum mechanics.* Springer, 2014.

[54] M. P. Bendsøe and N. Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71:197–224, 1988.

[55] M. P. Bendsøe. Optimal shape design as a material distribution problem. *Structural Optimization*, 1:193–202, 1989.

[56] S. J. Hollister and N. Kikuchi. A comparison of homogenization and standard mechanics analyses for periodic porous composites. *Computational Mechanics*, 10:73–95, 1992.

[57] M. P. Bendsøe and O. Sigmund. Material interpolation schemes in topology optimization. *Archive of applied mechanics*, 69:635–654, 1999.

[58] O. Sigmund. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33:401–424, 2007.

[59] B. Bourdin. Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50:2143–2158, 2001.

[60] T. E. Bruns and D. A. Tortorelli. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, 190:3443–3459, 2001.

[61] O. Sigmund and J. Petersson. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural Optimization*, 16:68–75, 1998.

[62] T. Borrvall. Topology optimization of elastic continua using restriction. *Archives of Computational Methods in Engineering*, 8:351–385, 2001.

[63] O. Sigmund and K. Maute. Sensitivity filtering from a continuum mechanics perspective. *Structural and Multidisciplinary Optimization*, 46:471–475, 2012.

[64] V. Braibant and C. Fleury. Shape optimal design using B-splines. *Computer Methods in Applied Mechanics and Engineering*, 44:247–267, 1984.

[65] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *ACM SIGGRAPH Computer Graphics*, 20:151–160, 1986.

[66] D. Chauhan, P. Chandrashekarappa, and R. Duvigneau. Wing shape optimization using ffd and twist parameterization. In *12th Aerospace Society of India CFD Symposium*, 2010.

[67] J. Samareh. Aerodynamic shape optimization based on free-form deformation. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4630, 2004.

[68] A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70:646–670, 2012.

[69] F. Ballarin, A. Manzoni, G. Rozza, and S. Salsa. Shape optimization by free-form deformation: existence results and numerical solution for stokes flows. *Journal of Scientific Computing*, 60:537–563, 2014.

[70] P. Schröder. Subdivision for modeling and animation. In *ACM SIGGRAPH*, 1998.

[71] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, 1978.

[72] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, 1978.

[73] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[74] U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin. Tuning subdivision by minimising gaussian curvature variation near extraordinary vertices. In *Computer Graphics Forum*, volume 25, pages 263–272. Wiley Online Library, 2006.

[75] C. De Boor, K. Höllig, and S. Riemenschneider. *Box splines.* Springer Science & Business Media, 2013.

[76] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 113–120. ACM Press/Addison-Wesley Publishing Co., 2000.

[77] M. A. Sabin and A. Bejancu. Boundary conditions for the 3-direction box-spline. In *Mathematics of Surfaces*, pages 244–261. Springer, 2003.

[78] N. M. Patrikalakis and T. Maekawa. *Shape interrogation for computer aided design and manufacturing.* Springer Science & Business Media, 2009.

[79] J. T. Klosowski, M. Held, J. S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization & Computer Graphics*, pages 21–36, 1998.

[80] D. F. Rogers and R. Earnshaw. *Computer graphics techniques: theory and practice.* Springer Science & Business Media, 2001.

[81] L. Piegl and W. Tiller. *The NURBS book.* Springer Science & Business Media, 2012.

[82] E. G. Houghton, R. F. Emnett, J. D. Factor, and C. L. Sabharwal. Implementation of a divide-and-conquer method for intersection of parametric surfaces. *Computer Aided Geometric Design*, 2:173–183, 1985.

[83] T. W. Sederberg, D. C. Anderson, and R. N. Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics, and Image Processing*, 28:72–84, 1984.

[84] C. M. Hoffmann. *Geometric and solid modeling.* Morgan Kaufmann, 1989.

[85] D. Manocha and J. F. Canny. A new approach for surface intersection. *International Journal of Computational Geometry & Applications*, 1:491–516, 1991.

[86] T. Dokken. *Aspects of intersection algorithms and approximation.* PhD thesis, University of Oslo, 1997.

[87] D. Cox, R. Goldman, and M. Zhang. On the validity of implicitization by moving quadrics for rational surfaces with no base points. *Journal of Symbolic Computation*, 29:419–440, 2000.

[88] L. Busé, D. Cox, and C. D'Andrea. Implicitization of surfaces in $\mathbb{P}^3$ in the presence of base points. *Journal of Algebra and its Applications*, 2:189–214, 2003.

[89] A. Khetan and C. D'Andrea. Implicitization of rational surfaces using toric varieties. *Journal of Algebra*, 303:543–565, 2006.

[90] C. Loop and S. Schaefer. Approximating catmull-clark subdivision surfaces with bicubic patches. *ACM Transactions on Graphics*, 27:8, 2008.

[91] P. Van Dooren. The computation of kronecker's canonical form of a singular pencil. *Linear Algebra and Its Applications*, 27:103–140, 1979.

[92] G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, fifth edition, 2016.

[93] N.A. Fleck, V.S. Deshpande, and M.F. Ashby. Micro-architectured materials: past, present and future. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 466, pages 2495–2516. The Royal Society, 2010.

[94] G. Gaël, J. Benoît, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[95] G. Hasle, K.-A. Lie, and E. Quak. *Geometric modelling, numerical simulation, and optimization.* Springer, 2007.

[96] I. Harari and E. Shavelzon. Embedded kinematic boundary conditions for thin plate bending by nitsche's approach. *International Journal for Numerical Methods in Engineering*, 92:99–114, 2012.

[97] I. Harari and E. Grosu. A unified approach for embedded boundary conditions for fourth-order elliptic problems. *International Journal for Numerical Methods in Engineering*, 104:655–675, 2015.

[98] S. Fernández-Méndez and A. Huerta. Imposing essential boundary conditions in mesh-free methods. *Computer Methods in Applied Mechanics and Engineering*, 193:1257–1275, 2004.

[99] V.S. Deshpande, M.F. Ashby, and N.A. Fleck. Foam topology: bending versus stretching dominated architectures. *Acta Materialia*, 49:1035–1040, 2001.

[100] J.B. Berger, H.N.G. Wadley, and R.M. McMeeking. Mechanical metamaterials at the theoretical limit of isotropic elastic stiffness. *Nature*, 543:533, 2017.

[101] Autodesk Inc. *Autodesk Simulation Mechanical: User Guide*, 2018.

[102] V.S. Deshpande and N.A. Fleck. Collapse of truss core sandwich beams in 3-point bending. *International Journal of Solids and Structures*, 38:6275–6305, 2001.

[103] H. G. Allen. *Analysis and design of structural sandwich panels.* Pergamon Press Ltd., 1969.

[104] O. Sigmund, N. Aage, and E. Andreassen. On the (non-) optimality of michell structures. *Structural and Multidisciplinary Optimization*, 54:361–373, 2016.

[105] M. Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[106] Z. Michalewicz. Evolution strategies and other methods. In *Genetic Algorithms+ Data Structures= Evolution Programs*, pages 159–177. Springer, 1996.

[107] A. Antoniou and W. Lu. *Practical optimization: algorithms and engineering applications*. Springer Science & Business Media, 2007.

[108] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[109] M. R.R.A. Joaquim and N. M.K. Poon. On structural optimization using constraint aggregation. In *VI World Congress on Structural and Multidisciplinary Optimization*, 2005.

[110] G. Kreisselmeier and R. Steinhauser. Systematic control design by optimizing a vector performance index. In *Computer aided design of control systems*, pages 113–117. Elsevier, 1980.

[111] V. S. Deshpande, N. A. Fleck, and M. F. Ashby. Effective properties of the octet-truss lattice material. *Journal of the Mechanics and Physics of Solids*, 49:1747–1769, 2001.

[112] J. Park and A. Sutradhar. A multi-resolution method for 3d multi-material topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 285:571–586, 2015.

[113] M. Paz and Leigh W. Static condensation and substructuring. In *Integrated Matrix Analysis of Structures*, pages 239–260. Springer, 2010.

[114] K. Bletzinger and E. Ramm. Form finding of shells by structural optimization. *Engineering with Computers*, 9:27–35, 1993.

[115] X. Xiao, M. Sabin, and F. Cirak. Interrogation of spline surfaces with application to isogeometric design and analysis of lattice-skin structures. *arXiv preprint arXiv:1810.07982*, 2018.

[116] T. B. Sercombe, X. Xu, V.J. Challis, R. Green, S. Yue, Z. Zhang, and P. D. Lee. Failure modes in high strength and stiffness to weight scaffolds produced by selective laser melting. *Materials & Design*, 67:501–508, 2015.

[117] D. Brackett, I. Ashcroft, and R. Hague. Topology optimization for additive manufacturing. In *Proceedings of the Solid Freeform Fabrication Symposium*, volume 1, pages 348–362. S, 2011.

[118] X. Wang, S. Xu, S. Zhou, W. Xu, M. Leary, P. Choong, M. Qian, M. Brandt, and Y. Xie. Topological design and additive manufacturing of porous metals for bone scaffolds and orthopaedic implants: a review. *Biomaterials*, 83:127–141, 2016.

[119] A. T. Gaynor and J. K. Guest. Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design. *Structural and Multidisciplinary Optimization*, 54:1157–1172, 2016.

[120] I. A. Roberts, C.J. Wang, R. Esterlein, M. Stanford, and D.J. Mynors. A three-dimensional finite element analysis of the temperature field during laser melting of metal powders in additive layer manufacturing. *International Journal of Machine Tools and Manufacture*, 49:916–923, 2009.

[121] J. Ding, P. Colegrove, J. Mehnen, S. Ganguly, S. P.M. Almeida, F. Wang, and S. Williams. Thermo-mechanical analysis of wire and arc additive layer manufacturing process on large multi-layer parts. *Computational Materials Science*, 50:3315–3322, 2011.

[122] M. Wallin, V. Jönsson, and E. Wingren. Topology optimization based on finite strain plasticity. *Structural and multidisciplinary optimization*, 54:783–793, 2016.

[123] M. Wallin, N. Ivarsson, and D. Tortorelli. Stiffness optimization of non-linear elastic structures. *Computer Methods in Applied Mechanics and Engineering*, 330:292–307, 2018.

[124] J. Stam. Evaluation of loop subdivision surfaces. In *SIGGRAPH'98 CDROM Proceedings*. Citeseer, 1998.

[125] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1995.

[126] K. Svanberg. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24:359–373, 1987.

[127] C. Fleury and V. Braibant. Structural optimization: a new dual method using mixed variables. *International Journal for Numerical Methods in Engineering*, 23:409–428, 1986.