© 2014 by Tomás Zegard. All rights reserved.

STRUCTURAL OPTIMIZATION: FROM CONTINUUM AND GROUND STRUCTURES TO ADDITIVE MANUFACTURING

BY

TOMÁS ZEGARD

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor Glaucio H. Paulino; Chair and Director of research
William F. Baker; Skidmore, Owings & Merrill LLP
Professor Paolo Gardoni
Dr. Arkadiusz Mazurek; Skidmore, Owings & Merrill, LLP
Professor Ivan Menezes; Pontifícia Universidade Católica do Rio de Janeiro
Professor Luke Olson

Abstract

This work focuses on optimal structural systems, which can be modeled using discrete elements (e.g. slender columns and beams), continuum elements (e.g. walls or slabs), or combinations of these. Optimization problems become meaningful only after the objective function, or benchmark, that evaluates a given design has been defined. Thus, it is logical to explore a variety of objectives, with emphasis on the ones that yield distinct results. The design may include constraints in response to performance or habitability, which must be included in the optimization to yield feasible designs.

Structural optimization can be used to improve structural designs by giving cheaper, stronger, lighter and safer structures. Gradient–based optimization is the preferred approach in this work, for it consciously improves a design using the gradient information, as opposed to making random guesses. The optimization problem has an internal dependency on structural analysis, which may require modifications or careful analysis, in order to obtain meaningful gradient information.

Simple problems composed solely of discrete elements are of particular interest to engineers in practice. The design of lateral bracing systems falls into this category. A novel discrete element topology optimization algorithm is proposed, and to facilitate the adoption by industry and academia, the implementation is also provided. Discrete element topology optimization has the potential to aid in the discovery of new closed–form solutions for common problems in structural engineering. These closed–form solutions, while often impractical to build, give insight into the physics of the optimal structural system. This information can be used to steer civil structural projects towards more efficient load transfer systems.

The manufacturing of optimal structures often lags behind our ability to analyse and design them. Additive manufacturing presents itself as the (much sought) final stage required for a complete structural optimization design process. A clean and streamlined methodology for manufacturing optimal structures is proposed. This includes optimal structures obtained from density–based methods as well as the ground structure method. The goal of this work is to improve the current sequential design process of civil structures. It does so by facilitating the integration of optimization techniques into existing design processes, in addition to extending optimization algorithms to address a wider variety of problems. Despite being centered primarily on civil structures, this work has the potential to impact other disciplines. In particular, an example that incorporates optimization techniques into the medical field is shown.

To my parents, Ketty & Gastón.

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Glaucio H. Paulino, for his help and providing me with the opportunity to conduct graduate studies. His constant encouragement to pursue novel, interesting and challenging problems are reflected in this document. I gratefully acknowledge the support of the Fulbright–CONICYT scholarship program, without whom I would have never been able to pursue this goal. I recognize the financial support of the National Science Foundation (NSF) for projects CMMI 1321661 and CMMI 1335160. In addition, I also acknowledge partial support from the Donald B. and Elizabeth M. Willett endowment at the University of Illinois at Urbana–Champaign.

I am very thankful to the collaboration opportunity with Skidmore, Owings & Merrill (SOM); for the feedback and constant stream of challenging and applied problems for me to explore. In particular, the guidance and insight of Bill Baker were essential to the transfer component of this research to industry. In addition, I owe a debt of gratitude to Arek Mazurek for his feedback, comments and good sense of humor. The words of encouragement and support from Professor Ivan Menezes in times when things did not work as expected, in addition to his feedback, were essential during my studies. I would like to acknowledge Professor Luke Olson; his vision from outside the field of structural engineering was very helpful and provided a much needed different perspective to problems. I would also like to thank Bill Baker, Professor Paolo Gardoni, Arek Mazurek, Professor Ivan Menezes and Professor Luke Olson for participating in my PhD defense committee.

I am very thankful of my friends. It would be impossible to name them all, but I would like to specially acknowledge Tatiana Afanasyeva, Beth Baumgartner, Lauren Beghini, Max

Bobrovskyy, Collin Carlier, Heng Chi, Pablo Farías, Roberto Jiménez, Sofie Leon, Maria Lobkis, Julián Marín, Daniel Maturana, Daniel Rubin, Max Silva, Daniel Spring, Cam Talischi and Guillermo Zañartu. Their support, help, humor and friendship made graduate school some of the best days of my life. I would also like to thank Emily Ewers for her unconditional support, patience and confidence.

Finally, my deepest gratitude and love are for my parents, Ketty and Gastón, and my brothers Christian and Gastón Andrés. Their never-ending support and confidence in me kept me going through these years away from home. My mother was influential in my belief in education and scholarship, and taught me to always give the best of myself in whatever it is that I do.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Currently, limited worldwide resources are driving research and development in all areas towards efficiency and optimality. Structural engineering is not exempt from this trend: new requirements for extreme and/or cheaper structures challenge the traditional design procedures.

The traditional building design sequence begins with a topology (usually) determined by the architect. This topology is then sized by the structural engineer in order for the project to materialize. Advances in material science, computational power and structural analysis have spoiled architecture by allowing for even more radical designs. With efficiency in mind, however, the design must go back to its roots where the shape was dictated not only by aesthetics but by structural behavior as well. Thus, the driving force behind an optimal design may be one, or a combination, of the following objectives (Figure 1.1): limited resources, extreme structural requirements and/or structural functionality (safe structures).

The work presented in this document, explores a variety of topics and areas of optimal structural design: computer algorithms, design tools, structural optimization formulations, discrete-continuum element optimization and additive manufacturing. The ultimate goal is to make tangible contributions in these areas to be incorporated in future structural systems.

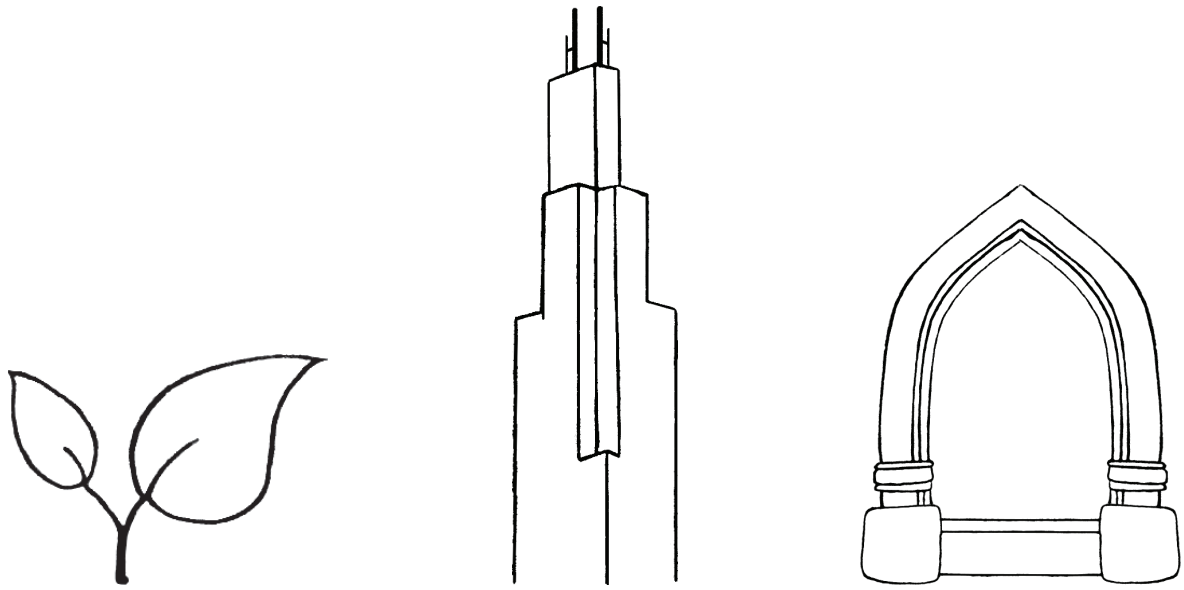


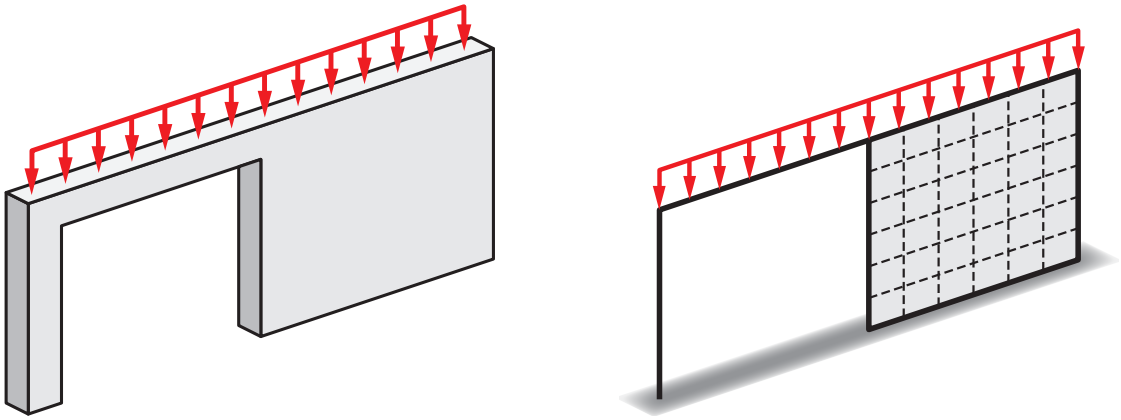
Figure 1.1: The driving forces behind structural optimization. Projects may decide to include structural optimization motivated by these concepts.

1.1 Motivation

1.1.1 Mixed discrete–continuum optimization

Structural modeling and analysis consist of the harmonious linkage of multi–scales, materials and shapes. Some information is implicitly represented in the computer model (e.g. steel reinforcement within the concrete), while other must be explicitly present as an element itself (e.g. a column or a beam). These explicit elements may be of discrete or continuum type. Depending on the element's dimension and the level of detail of the model, a column may be modeled as an infinitely thin member with representative properties, whereas a wall may be modeled as a *flat* continuum (Figure 1.2).

Structural optimization often deals with the optimization of discrete or continuum structures, but a combination of both is rare. Only recently, continuum optimization with discrete elements has gained some attention (Allahdadian et al., 2012; Liang et al., 2000; Liang, 2007;



(a) (b)

Figure 1.2: Example of a model using discrete and continuum elements. (a) Column–Beam–Wall frame. (b) Idealized model for the column–beam–wall using discrete and continuum elements.

Mijar et al., 1998; Stromberg et al., 2012), and the same can be said of discrete optimization (truss layout) in the presence of a continuum (Amir and Sigmund, 2013; Kato and Ramm, 2010; Zegard and Paulino, 2013b). Nonetheless, current developments often suffer from limitations and gaps in their application, as is expected from new developments. Moreover, the simultaneous optimization of both, discrete and continuum elements is still unresolved.

1.1.2 Closed–form solutions for applied problems

Despite the fact that structural optimization has been in development for years (Topping, 1983), the library of known closed–form optimal solutions is restricted to relatively *simple* problems. In other words, these solutions have limited use in real civil structures. Presently, there are no known solutions for a variety of problems that could potentially impact the field of structural engineering (Figure 1.3): the optimal bracing pattern for a building subjected to wind loads; the optimal layout of the reinforcement in a beam; the optimal location for a suspension cable support on a cantilever beam, to name a few.

Numerical methods can provide *close enough* solutions for problems with no known analytical closed–form solutions. For all practical purposes, coarse numerical solutions are

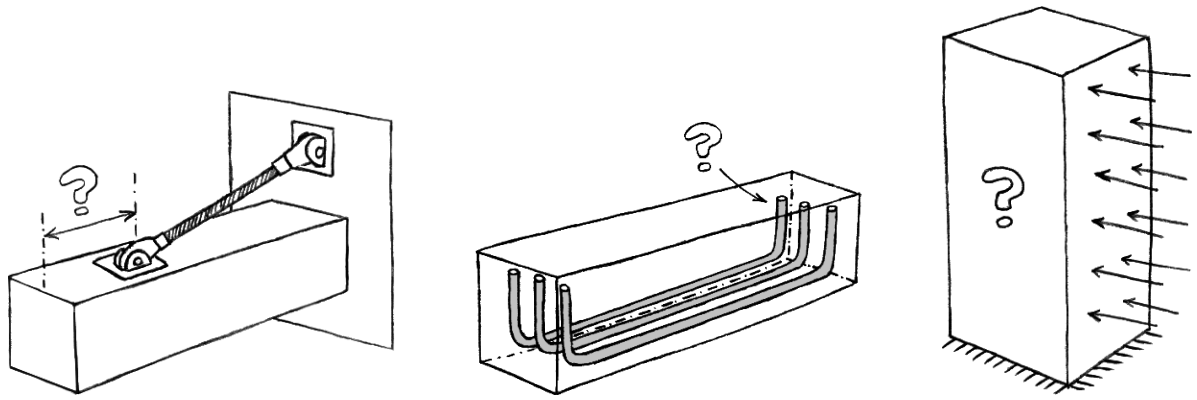
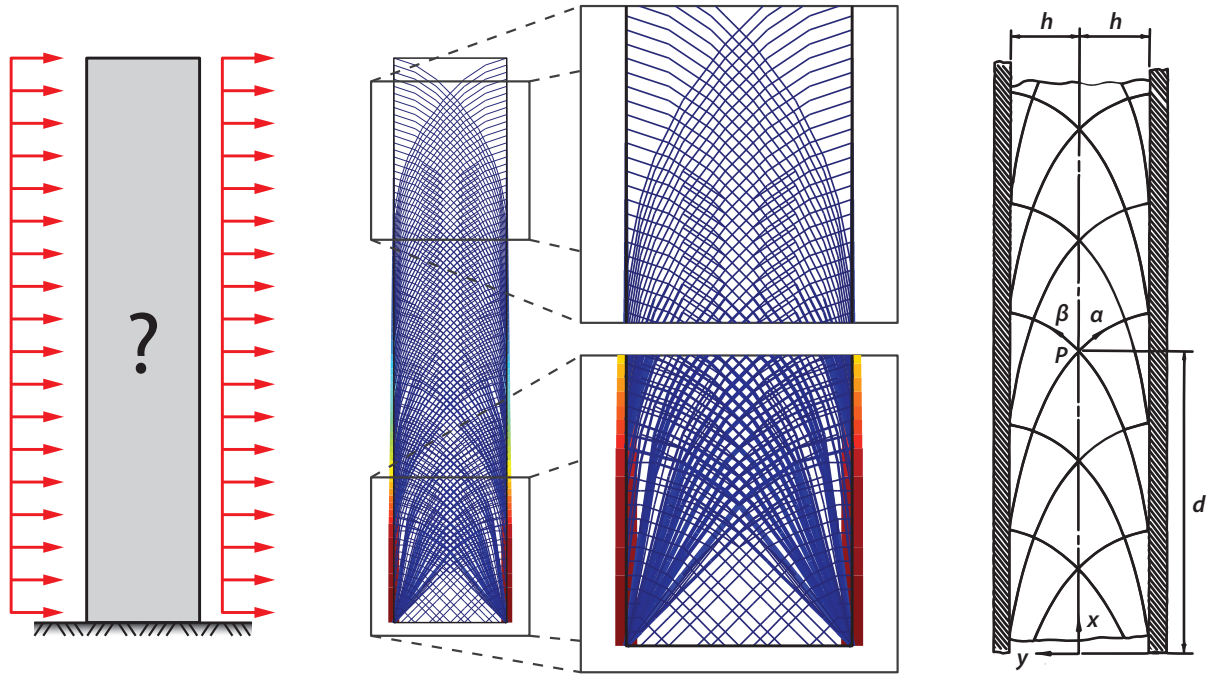


Figure 1.3: Structural problems with no known closed–form solutions. (a) Find the optimal cable anchor location for a cantilever beam. (b) Find the optimal reinforcement thickness, shape and spatial distribution for a beam. (c) Building loaded laterally by wind.

sufficiently accurate when contrasted with our ability to manufacture and specify structural designs. However, numerical solutions with a high degree of detail can aid in the development of new closed–form solutions. These in turn provide a structural benchmark and provides insight into the optimal structural system for the problem. Thus, the importance of highly detailed solutions is justified. The family of known closed–form analytical solutions began with the work of Michell (Michell, 1904; Hemp, 1973). Additional closed–form solutions and extensions can be found in the works of Rozvany and Gollub (1990); Lewiński et al. (1994b,a); Rozvany et al. (1997); Rozvany (1998); Lewiński (2004); Graczykowski and Lewiński (2005, 2006a,b,c, 2007); Lewiński and Rozvany (2007, 2008b,a); Lewiński et al. (2013), among others.

A sample workflow for obtaining closed–form solutions is as follows: Consider the problem in Figure 1.4(a) for example. The approximate discrete–element optimal solution is shown in Figure 1.4(b). The solution away from the supported base is related to the slip–line field (theory of plasticity) for a plate subjected to compression. There is thus a connection between the fields of structural optimization and theory of plasticity (Michell, 1904; Hencky, 1923). Further analysis of the problem in Figure 1.4 may result in a closed–form



(a) (b) (c)

Figure 1.4: Two–dimensional simplification of the problem of a building subjected to lateral loads. (a) Domain, loading and supports. (b) Approximated optimal structure solution obtained using the algorithm and implementation described in Chapter 4 and Appendix A. (c) Slip–line field for a sufficiently wide block compressed between perfectly rough platens (Chakrabarty, 2006).

analytical solution for building bracing patterns. Nonetheless, the solution in Figure 1.4(b) is sufficiently detailed to serve as the starting point of an efficient building design.

1.2 Document outline and organization

The remainder of this document is organized as follows: Chapter 2 presents a methodology to couple discrete and continuum structural elements such that a gradient–based optimization for the discrete elements can be applied. In Chapter 3, we employ a variety of optimization algorithms and formulations to the problem of finding the optimal bracing system for building loaded laterally. Chapter 4 describes in detail a two–dimensional ground–structure based topology optimization implementation for concave domains and with the possibility

of holes. The algorithm is extended to three–dimensional space in Chapter 5. Both the two– and three–dimensional implementation and algorithms (Chapters 4 and 5), are verified and compared against known closed–form solutions. Chapter 6 describes the general procedure to manufacture and integrate results obtained from topology optimization into structural designs. The conclusions from each chapter are aggregated and summarized in Chapter 7, along with potential extensions of the present work. Finally, educational implementations in MATLAB are provided in the Appendices A and B. This work has resulted in a number of publications in peer–reviewed journals, and thus the reader is also referred to Zegard and Paulino (2013b); Zegard et al. (2014); Zegard and Paulino (2014a,b).

Chapter 2

Truss layout optimization within a continuum

Several methods exist for structural optimization; unfortunately none of them is able to tackle every problem, and all have drawbacks. The most popular optimization methods consist of optimizing a density material distribution in a continuum (Bendsøe and Sigmund, 2003), optimizing a truss layout (Felix and Vanderplaats, 1987; Hansen and Vanderplaats, 1990; Lipson and Gwin, 1977; Ohsaki, 2010), and optimizing the shape of a continuum (Haslinger and Mäkinen, 2003). Truss layout optimization has greatly evolved with the *ground structure* method (Dorn et al., 1964; Sokół, 2011), and proves to be a reliable and robust method for the optimization of truss structures. However, the optimization of a structure that combines both discrete and continuum elements has many caveats today. Material distribution of a continuum with an overlaying truss structure has been previously studied (Allahdadian et al., 2012; Liang et al., 2000; Liang, 2007; Mijar et al., 1998), and recent refinements make it suitable for real applications (Stromberg et al., 2012). Previously, a formulation for embedding reinforcement (discrete elements) in the context of reinforced concrete was developed (Elwi and Hrudey, 1989), and later extended to three-dimensions (Barzegar and Maddipudi, 1994). Optimization of reinforced concrete using this embedded formulation was also explored (Kato and Ramm, 2010). The ground-structure method with elements embedded in a continuum has also proven to be feasible (Amir and Sigmund, 2013). This work attempts to solve the problem where discrete structure, linked to a continuum

(or embedded), is geometrically optimized. In this process, the discrete nodes will not directly match over continuum nodes, and a convolution-based coupling was developed. Some examples of structures typically modeled in a discrete-continuum fashion are: reinforced concrete, cable supported bridges, column supporting a slab and beam-wall connections to name a few.

If the continuum is modeled using traditional C^0 elements, the first derivatives of the displacement field are discontinuous, thus making the embedded formulation difficult to optimize using traditional gradient based optimizers. The discontinuity problem could potentially be solved using C^1 elements, however, the formulations for these are complex, especially for higher dimensions. Thus, the goal is to develop a formulation to couple discrete truss elements to continuum elements in a simple way, yet sophisticated enough to obtain a smooth derivative field necessary for gradient based optimizers. An example of a situation that requires such framework is given in Figure 2.1, and consists of a simply-supported deep beam with cable supports loaded by self-weight. The problem consists of optimizing the anchor location for the cable supports.

This formulation is based on small deformation theory, and because nodes are treated as a cloud, any type or order of finite elements can be used (i.e. the element connectivity is not used). The examples in the present work deal with compliance (external work) minimization. Nevertheless, the technique can be applied to any objective function based on stiffness for which an expression for the gradient can be obtained.

2.1 Formulation

Truss layout optimization has been explored previously with good results (Felix and Vanderplaats, 1987; Hansen and Vanderplaats, 1990; Lipson and Gwin, 1977; Ohsaki, 2010). The formulation for truss layout optimization presented here is analogous to the one presented in (Hansen and Vanderplaats, 1990), but better suited for any-dimensional (1D, 2D, 3D)

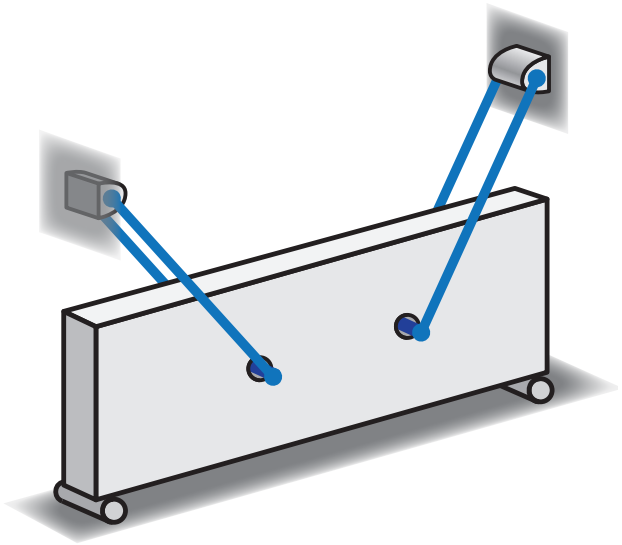


Figure 2.1: Simply-supported deep beam with cable supports loaded by self-weight. Discrete—continuum structural optimization can provide the optimal anchor point locations for the cable supports.

problems and extended by combining it with a continuum.

The stiffness matrix for a truss element in local coordinates is

$$\mathbf{K}^{\star} = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{2.1}$$

with A, E and L being the element's cross-sectional area, Young modulus, and length respectively. Given the truss element schematic in Figure 2.2, the directional cosines vector $\mathbf{d}$ is defined as

$$\begin{aligned} \mathbf{d}_{2D} &= \frac{1}{L}\left[x_2 - x_1, y_2 - y_1\right] \\ \mathbf{d}_{3D} &= \frac{1}{L}\left[x_2 - x_1, y_2 - y_1, z_2 - z_1\right] \ , \end{aligned} \tag{2.2}$$

and the transformation matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{d} & \mathbf{0} \\ \mathbf{0} & \mathbf{d} \end{bmatrix} \tag{2.3}$$

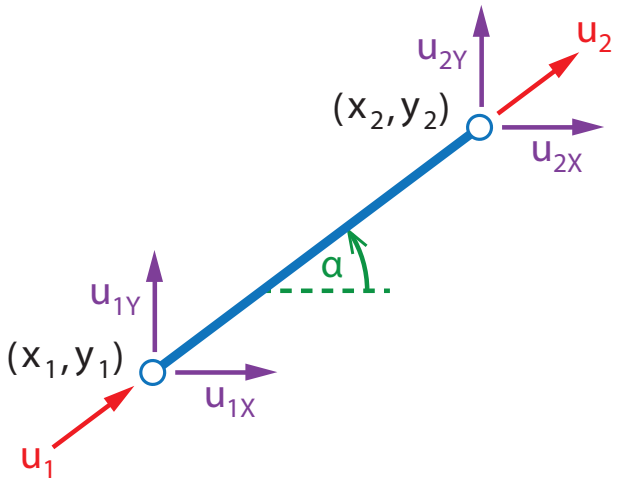


Figure 2.2: Two-dimensional truss element with local and global degrees-of-freedom and nodal coordinates (x_1, y_1) and (x_2, y_2).

The local degrees-of-freedom u_1 and u_2 can be related to the global DOFs as:

$$\left\{ \begin{array}{c} u_1 \\ u_2 \end{array} \right\} = \mathbf{T}_{2D} \left\{ \begin{array}{c} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \end{array} \right\} \quad , \quad \left\{ \begin{array}{c} u_1 \\ u_2 \end{array} \right\} = \mathbf{T}_{3D} \left\{ \begin{array}{c} u_{x1} \\ u_{y1} \\ u_{z1} \\ u_{x2} \\ u_{y2} \\ u_{z2} \end{array} \right\} \tag{2.4}$$

The stiffness matrix in global coordinates $\mathbf{K}_e$ for truss element e in terms of the local stiffness $\mathbf{K}_e^\star$ and the transformation matrix $\mathbf{T}_e$ is

$$\mathbf{K}_e = \mathbf{T}_e^T \mathbf{K}_e^\star \mathbf{T}_e \tag{2.5}$$

The derivative of the global stiffness matrix with respect to the coordinate n of node j

of the truss member is

$$\frac{\partial \mathbf{K}_e}{\partial n_j} = \frac{\partial \mathbf{T}_e^T}{\partial n_j}\mathbf{K}_e^\star \mathbf{T}_e + \mathbf{T}_e^T \frac{\partial \mathbf{K}_e^\star}{\partial L}\frac{\partial L}{\partial n_j}\mathbf{T}_e + \mathbf{T}_e^T \mathbf{K}_e^\star \frac{\partial \mathbf{T}_e}{\partial n_j} \tag{2.6}$$

with L representing the truss element's length, $n = \{x, y, z\}$ and $j = \{1, 2\}$. The derivatives of the element's length L, with respect to the coordinate n, are

$$\begin{aligned} \frac{\partial L}{\partial n_1} &= -d_n \\ \frac{\partial L}{\partial n_2} &= \quad d_n \end{aligned} \tag{2.7}$$

and the derivative of the stiffness matrix with respect to the element length is

$$\frac{\partial \mathbf{K}^\star}{\partial L} = -\frac{AE}{L^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{2.8}$$

The Jacobian matrix of the directional cosine vector with respect to the coordinates of the first truss element node ($j = 1$) is

$$\mathbf{J}_{(1)}(\mathbf{d}) = \frac{1}{L}\left(\mathbf{d}^T\mathbf{d} - \mathbf{I}\right) \tag{2.9}$$

and $\mathbf{J}_{(2)}(\mathbf{d}) = -\mathbf{J}_{(1)}(\mathbf{d})$. Inspecting a couple of terms, we obtain, for example:

$$\begin{aligned} \left[\mathbf{J}_{(1)}(\mathbf{d})\right]_{21} &= \frac{\partial \mathbf{d}_2}{\partial x_1} = \quad \frac{d_x d_y}{L} \\ \left[\mathbf{J}_{(2)}(\mathbf{d})\right]_{22} &= \frac{\partial \mathbf{d}_2}{\partial y_2} = -\frac{d_y d_y - 1}{L} \end{aligned} \tag{2.10}$$

and with this the derivatives of the transformation matrix $\mathbf{T}$ are completely defined. This formulation is equivalent to the one presented in (Hansen and Vanderplaats, 1990), but better suited to be coupled with a continuum.

2.1.1 Mapping discrete to continuum

Consider the (global) stiffness matrix of a continuum $\mathbf{K}_c$ obtained by means of a finite element method (FEM), and the stiffness matrix from a single truss element $\mathbf{K}_e$ in global coordinates. The challenge is to add the contribution of $\mathbf{K}_e$ onto $\mathbf{K}_c$ in a coherent fashion (energy conservation), and with a smooth derivative field. An approach based on energy conservation and FEM shape functions meets the first requirement, but because the FEM shape functions are discontinuous across elements, it does not have a smooth derivative field.

The components of the truss' stiffness matrix $\mathbf{K}_e$ associated with a single node will be mapped to another matrix $\mathbf{K}_e^+$ in terms of the continuum nodes, so that its contribution can be added to $\mathbf{K}_c$. Calling the DOFs associated with the truss node $\mathbf{u}$ and the ones from the continuum $\mathbf{u}_c$, the mapping based on energy conservation is:

$$\mathbf{u}^T\mathbf{K}_e\mathbf{u} = \mathbf{u}_c^T\mathbf{K}_e^+\mathbf{u}_c \tag{2.11}$$

Because the truss node is within the continuum, the displacement field at the truss' node location is an interpolation of the values in the continuum at known discrete positions. Using some shape function $\mathbf{N}$ to interpolate the continuum field, the degrees of freedom of the continuum and truss's node are related as:

$$\mathbf{u} = \mathbf{N}\mathbf{u}_c \ , \tag{2.12}$$

and thus Equation (2.11) becomes:

$$\begin{aligned}
\mathbf{u}^T\mathbf{K}_e\mathbf{u} &= \mathbf{u}_c^T\mathbf{K}_e^+\mathbf{u}_c \\
\left(\mathbf{N}\mathbf{u}_c\right)^T\mathbf{K}_e\left(\mathbf{N}\mathbf{u}_c\right) &= \mathbf{u}_c^T\mathbf{K}_e^+\mathbf{u}_c \\
\mathbf{u}_c^T\left(\mathbf{N}^T\mathbf{K}_e\mathbf{N}\right)\mathbf{u}_c &= \mathbf{u}_c^T\mathbf{K}_e^+\mathbf{u}_c \\
\mathbf{N}^T\mathbf{K}_e\mathbf{N} &= \mathbf{K}_e^+
\end{aligned} \tag{2.13}$$

The mapping described in Equation (2.13) is done for every truss node being mapped to the continuum. If traditional FEM shape functions are used in $\mathbf{N}$, the derivative of the mapped stiffness with respect to the truss nodal position becomes problematic due to discontinuities in the shape function derivatives across elements. In detail:

$$\frac{\partial \mathbf{K}_e^+}{\partial n_j} = \frac{\partial \mathbf{N}^T}{\partial n_j}\mathbf{K}_e\mathbf{N} + \mathbf{N}^T\frac{\partial \mathbf{K}_e}{\partial n_j}\mathbf{N} + \mathbf{N}^T\mathbf{K}_e\frac{\partial \mathbf{N}}{\partial n_j} \ , \tag{2.14}$$

while the second term in Equation (2.14) is smooth throughout the whole continuum, the first and third terms are not. In practical applications, the discontinuities increase with the number of elements: more elements result in more edges, and therefore more discontinuities. For highly refined continuum meshes, there will be a high number of local minima close to the optimum. This situation will prevent the optimizer from converging to the true (ideally global) optimum.

The choice of the shape functions $\mathbf{N}$ used in the mapping to $\mathbf{K}_e^+$ is of critical importance to obtain an embedded formulation with a smooth gradient field. In addition to the inter-element discontinuity, the truss node position needs to be mapped into the parent element coordinates (typically xi, eta and $zeta$), if an isoparametric formulation is to be used, as is the case for previous embedded formulations (Elwi and Hrudey, 1989; Barzegar and Maddipudi, 1994). The alternative proposed in the present work is to use shape functions based on a convolution operator. These can be arbitrarily smooth up to any derivative depending on the convolution function chosen (although we are only interested in the first derivative), and do not need to be mapped to parent coordinates since they operate in the actual node coordinates of the continuum.

2.1.2 Convolution-based shape functions

Sacrificing some coherence in the coupling (different shape functions used to analyze the continuum and for the embedding), an approach based on a convolution operator is pro-

posed. This approach consists of representing the truss DOFs as a convolution of the nearby continuum nodes. That is, we use a shape function $\tilde{\mathbf{N}} \neq \mathbf{N}$, with $\tilde{\mathbf{N}}$ built from a convolution operator $h(\cdot)$, that ensures smoothness of the gradient field by complying with

$$\begin{aligned} h(0) &= 1 \\ h(r \geq R) &= 0 \\ \left.\frac{dh}{dr}\right|_{r=R} &= 0 \end{aligned} \tag{2.15}$$

with R defined as the convolution operator radius, and r the distance between the truss and continuum node. In addition, the shape functions $\tilde{\mathbf{N}}$ must preserve partition of unity

$$\sum_k \tilde{\mathbf{N}}_k = 1 \tag{2.16}$$

Two possible functions for $h(\cdot)$ are presented in Equation (2.17), but any other function that complies with Equation (2.15) can be used

$$\begin{aligned} h_1(r) &= \begin{cases} 1 - \sin\left(\frac{r\pi}{2R}\right) & r \leq R \\ 0 & r > R \end{cases} \\ h_2(r) &= \begin{cases} \left(\frac{r}{R}\right)^2 - 2\left(\frac{r}{R}\right) + 1 & r \leq R \\ 0 & r > R \end{cases} \end{aligned} \tag{2.17}$$

The functions presented in Equation (2.17) are plotted in Figure 2.3. The shape function for a truss node, associated with the continuum node a is

$$\tilde{N}_a = \frac{h(r_a)}{\sum_k h(r_k)} \tag{2.18}$$

The shape function derivative for a specific truss node corresponding to a continuum node

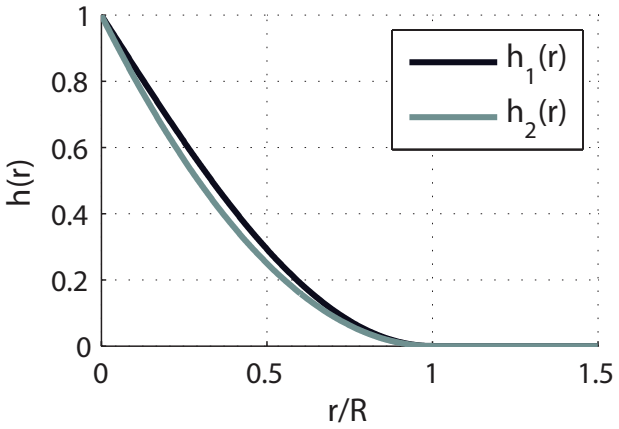


Figure 2.3: Plots of the convolution functions presented in Equation (2.17).

a with respect to coordinate n is

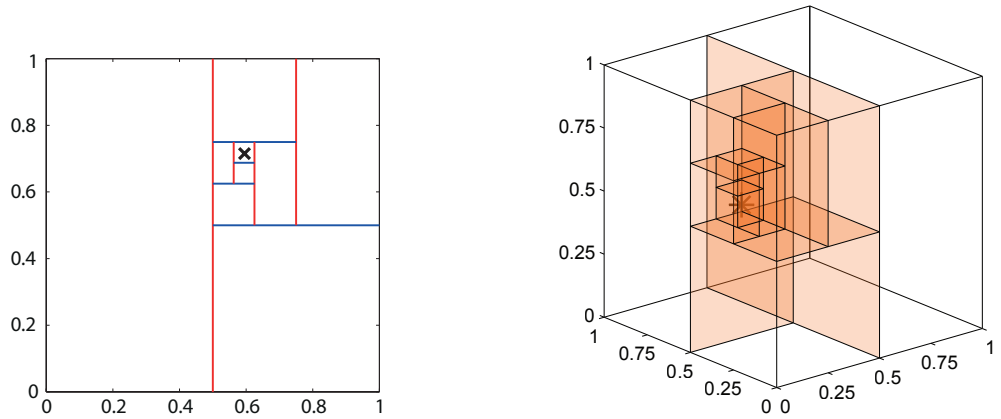
$$\frac{\partial \tilde{N}_a}{\partial n} = \frac{\frac{\partial h}{\partial n}(r_a)\sum_k h(r_k) - h(r_a)\sum_k \frac{\partial h}{\partial n}(r_k)}{\left[\sum_k h(r_k)\right]^2} \tag{2.19}$$

with the derivatives of the convolution functions as follows:

$$\begin{aligned} \frac{\partial h_1}{\partial n}(r) &= \begin{cases} \frac{\pi}{2R}\cos\left(\frac{r\pi}{2R}\right)\tilde{d}_n & r \le R \\ 0 & r > R \end{cases} \\ \frac{\partial h_2}{\partial n}(r) &= \begin{cases} -2\frac{r-R}{R^2}\tilde{d}_n & r \le R \\ 0 & r > R \end{cases} \end{aligned} \tag{2.20}$$

where $\tilde{\mathbf{d}}$ in this case is the directional cosine from the truss node to the continuum node (associated with the distance r). The sum in the denominator is through all the nodes in the continuum, but because the convolution function is zero for $r > R$, the sum only encompasses a few of the total nodes. The continuum nodes that fall within the convolution operator are found using a tree data structure, making the search for different truss nodes linking to continuum efficient. This tree scheme becomes a quadtree and an octtree in two and three dimensions respectively (Figure 2.4).

The convolution shape functions lack desirable properties like the Kronecker delta prop-



(a) (b)

Figure 2.4: Binary domain partition examples for 0—1 domains. (a) Quadtree in two dimensions (4 partitions, $P = [0.595 \quad 0.715]$. (b) Octtree in three dimensions (3 partitions, $P = [0.19 \quad 0.43 \quad 0.56]$.

erty ($\delta_{ii} = 1$ and $\delta_{ij} = 0$ for nodes $i \neq j$), because these shape functions are not associated to a specific node as with FEM shape functions, but to a cloud of nodes instead. However, it does comply with partition of unity (Equation (2.16)) and has no negative values anywhere. These convolution shape functions posses continuous first derivative field, a desirable property and required for the present work.

The mapping of $\mathbf{K}_e$ onto the continuum follows the energy conservation mapping described in Equation (2.13), but using $\tilde{\mathbf{N}}$ instead of the FEM shape functions. This is also analogous for Equation (2.14) resulting in the following expressions:

$$\mathbf{K}_e^+ = \tilde{\mathbf{N}}^T \mathbf{K}_e \tilde{\mathbf{N}} \tag{2.21}$$

$$\frac{\partial \mathbf{K}_e^+}{\partial n_j} = \frac{\partial \tilde{\mathbf{N}}^T}{\partial n_j} \mathbf{K}_e \tilde{\mathbf{N}} + \tilde{\mathbf{N}}^T \frac{\partial \mathbf{K}_e}{\partial n_j} \tilde{\mathbf{N}} + \tilde{\mathbf{N}}^T \mathbf{K}_e \frac{\partial \tilde{\mathbf{N}}}{\partial n_j} \tag{2.22}$$

Note that the dimensionality of $\tilde{\mathbf{N}}$ is variable and does not necessarily match with $\mathbf{N}$.

2.1.3 Connection with blur filters

This coupling to the continuum works by smearing (or blurring) the displacement field around the truss member node. Provided that the convolution radius is not too big, the error introduced by this method is controllable and provides a smooth derivative field throughout the continuum. The smearing error will have a higher impact when closer to a rapid variation of the field (i.e. sharp edges, single node loads and boundary conditions). There are reasons, however, that justify the application of the convolution and introduction of the error; the derivative field should be continuous, and the fields should be smooth (except for the aforementioned singularity points).

The convolution-based shape functions $\tilde{\mathbf{N}}$ operate in a similar fashion to blur filters used in image processing: blur filters apply a convolution function to the image. In the present work however, instead of blurring an image, the convolution *blurs* the displacement field to ensure continuous first derivatives. There are many blur filters used in image processing, with the *Gaussian blur* being one of the most common ones (Nixon and Aguado, 2012). Figure 2.5 applies the traditional gaussian blur and a convolution based on $h_2(r)$ to an image for comparison purposes. From inspection of the images and function plots it can be concluded that:

- The effect of the convolution based on $h_2(r)$ is more localized compared to the Gaussian blur. This is due to the rapid decay of the $h_2(r)$ function.
- The convolution of a Gauss function extends to infinity (most image processors ignore values beyond 6σ, with σ typically taken as half of the filter radius r). Thus, the convolution of $h_2(r)$ involves a smaller set of nodes.
- Using similar radii, the convolution based on $h_2(r)$ introduces a smaller distortion of the original data. This is again due to the rapid decay of this function.
- The convolution based on $h_2(r)$ allows large radii without great impact on the data.

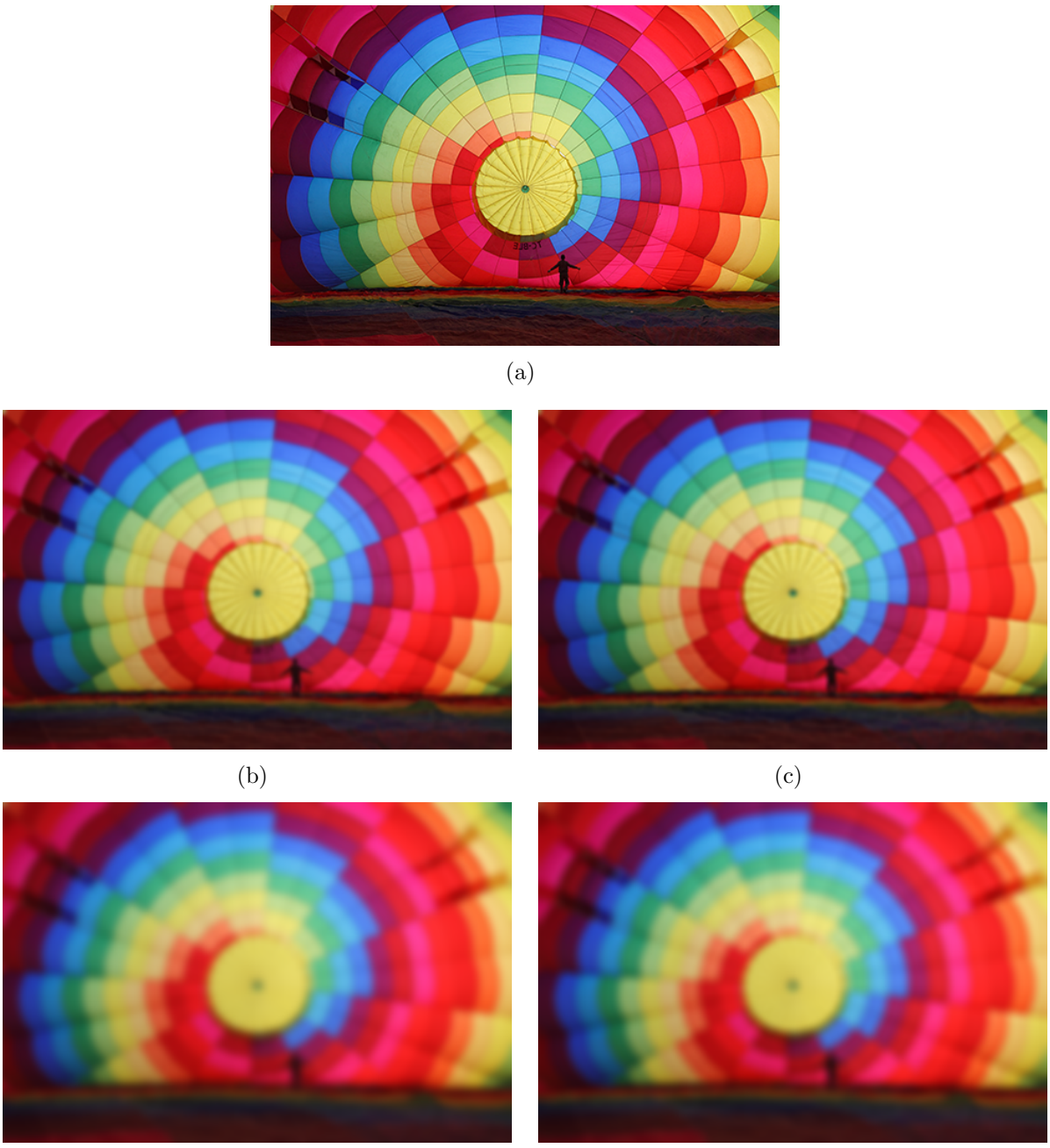


Figure 2.5: Image convolution comparison: (a) Original image [© Benh Lieu Song | licensed under CC-BY-SA-3.0] of size 480 × 320. (b) Gaussian blur with a 2 pixel radius. (c) Convolution with $h_2(r)$ with a 25 pixel radius. (d) Gaussian blur with a 4 pixel radius. (e) Convolution with $h_2(r)$ with a 100 pixel radius.

2.1.4 Optimization issues

The algorithm becomes unstable for a large number of variables if the variables are allowed to freely vary at each iteration. This is specially true for large number of variables. Thus, a move limit m enforces small variations from one iteration to the next. This results in a more cautious progression towards the optimum, and with the step size controlled by the move limit m, as follows:

$$\left|n_i^{new} - n_i^{old}\right| \leq m \quad \forall\ n = x, y, z \text{ and } i = 1 \ldots N_{nodes} \tag{2.23}$$

The move limit or variable bounds are common features in optimizers, making the implementation of Equation (2.23) simple.

The optimizer could decide to overlap two nodes together, typically resulting in a super-member (two members overlapping). Nevertheless, this might also result in a member of length $L = 0$, causing problems in Equations (2.1), (2.2), (2.8) and (2.9). To prevent this situation, a minimum length constraint $L_{\min} > 0$ for every member is included:

$$L_{\min} - L_e \leq 0 \tag{2.24}$$

Maximum truss volume $V_{\max}$ can be also specified as

$$\sum_e L_e A_e - V_{\max} \leq 0 \tag{2.25}$$

with the derivatives for the constraints in Equations (2.24) and (2.25) completely defined using Equations (2.8) and (2.7). If the element cross-sectional areas are also design variables, the derivative is trivial since

$$\frac{\partial \mathbf{K}_e^+}{\partial A_e} = \frac{E}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{2.26}$$

Setting a lower limit on the cross-sectional area is often recommended for stability and

constructibility reasons. In addition a move limit on the cross-sectional areas stabilizes the problem (analogous to Equation 2.23). These requirements translate into the following equations

$$\begin{aligned} A_e &> A_{\min} \\ \left|A_e^{new} - A_e^{old}\right| &\leq m_a \quad \forall e = 1 \dots N_{elems} \end{aligned} \tag{2.27}$$

2.2 Verification of the method

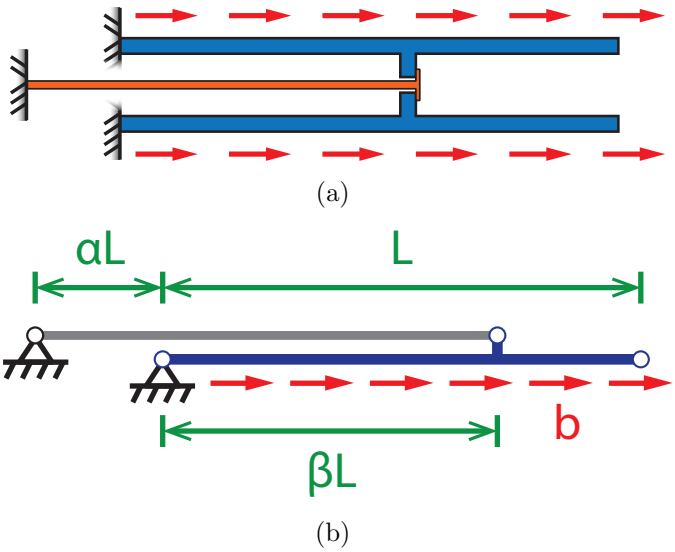
2.2.1 One-dimensional bar with a cable anchor

This problem seeks to find the optimal anchor position of a reinforcing cable within a weaker bar modeled as a continuum subjected to body force, as exemplified by Figure 2.6(a). The objective function for minimization is the compliance (external work) of the total structure (continuum and discrete). This model can be idealized as in Figure 2.6(b): αL is defined as the anchor point distance and βL is the anchor point measured within the continuum bar. The ratio between the bar (continuum) and cable stiffness is defined as $\gamma = EA/E_cA_c$, where E_c and A_c are the bar's Young modulus and cross-sectional area of the continuum, and E and A are the same but for the anchor cable. The design variable is the anchoring distance βL. This problem is of particular interest because an analytical solution can be obtained. The compliance of a single bar problem of length L, subjected to body force b and an end force P as in Figure 2.7 is:

$$C = \int_0^L u\left(x\right) b \, \mathrm{d}x + Pu\left(L\right) = \frac{1}{EA}\left(\frac{b^2L^3}{3} + PbL^2 + P^2L\right) \tag{2.28}$$

The displacement at the anchor point u_{anchor} can be obtained by simple structural analysis:

$$u_{anchor} = \frac{bL^2}{2AE}\frac{\beta\left(\alpha+\beta\right)\left(2-\beta\right)}{\alpha+\beta+\gamma\beta} \tag{2.29}$$



(a)

(b)

Figure 2.6: One-dimensional truss-continuum problem. (a) Bar (continuum) reinforced by a stiff cable (truss). (b) Idealized model of the bar with reinforcing cable.

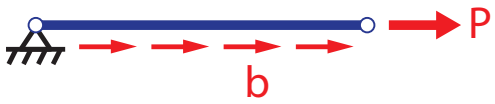


Figure 2.7: Simple model of a continuum subjected to a body force and a load at the tip.

The problem can be partitioned at the anchor point, and the expression in Equation (2.28) can be used for both segments of the continuum and the cable. The end force P taken by the bar segment of length βL is

$$P = \frac{bL}{2} \frac{2\alpha + 2\beta - 2\alpha\beta - 2\beta^2 - \gamma\beta^2}{\alpha + \beta + \gamma\beta} \tag{2.30}$$

Finally, the compliance for the complete problem is

$$C = \frac{b^2 L^3}{12EA} \frac{4\alpha + 4\beta + 4\gamma\beta - 12\gamma\beta^2 + 12\gamma\beta^3 - 3\gamma\beta^4}{\alpha + \beta + \gamma\beta} \tag{2.31}$$

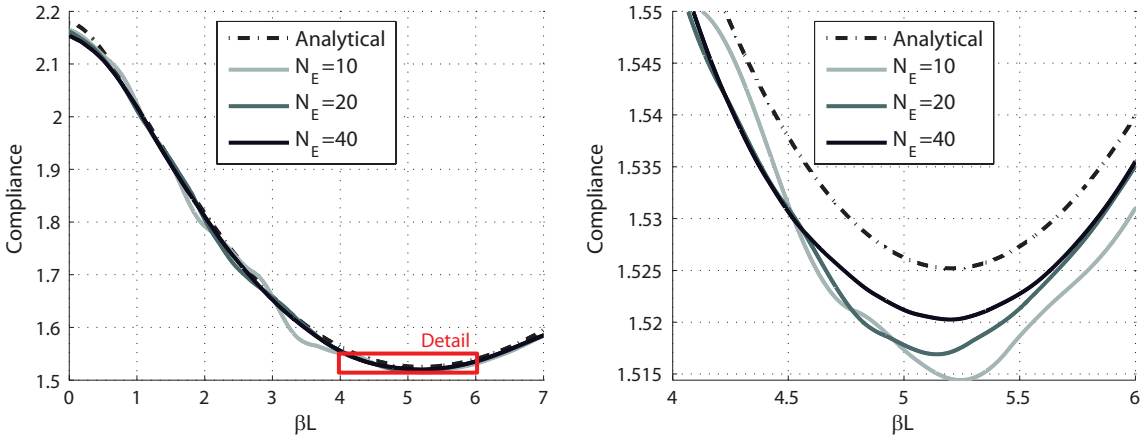


Figure 2.8: (a) Compliance with convolution coupling for different mesh refinements. (b) Detail close to the optimum.

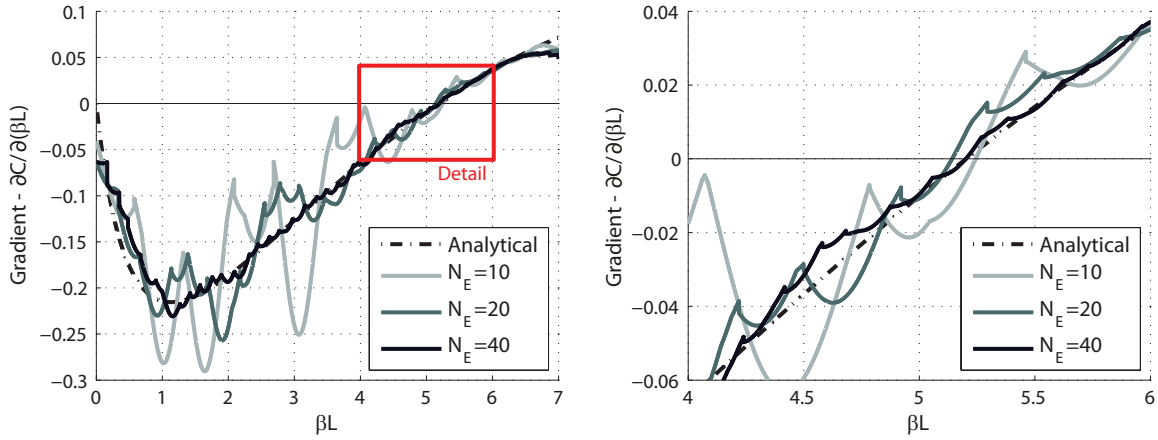
The optimal anchor point (minimizes compliance) is located at $\beta_c L$, with

$$\beta_c = \min\left(\frac{1+\gamma-2\alpha+\sqrt{1+8\alpha+2\gamma+4\alpha^2+8\alpha\gamma+\gamma^2}}{3+3\gamma}\,,\ 1\right) \tag{2.32}$$

Given the following problem data: $L = 7$, $\alpha = 2/7$, $E_c A_c = 210$, $EA = 150$, $\gamma = 5/7$ and $b = 2$, the embedding technique is performed for three different discretizations keeping the convolution radius fixed at $R = 0.2L$, and then compared to the analytical solution in Figure 2.8. The convolution function used is $h_2(\cdot)$ from Equation (2.17). The gradient is also compared in Figure 2.9. The optimal location is at $\beta_c = \left(2+\sqrt{22}\right)/9$ and $C(\beta_c) = 1.5252$. To ensure the algorithm is robust, the finite element size Δx is distributed randomly between $0.7L/N_E \leq \Delta x \leq 1.3L/N_E$, with N_E representing the number of elements of the partition. The minima for all meshes are presented in Table 2.1.

The same analysis is repeated keeping the mesh refinement fixed at $N_E = 20$ and varying the size of the convolution radius R. The results are compared in Figure 2.10. The gradient is also compared in Figure 2.11 and the minima for each case are presented in Table 2.2.

Using a mesh with evenly spaced elements, Figure 2.12 compares the analytical sensitivity

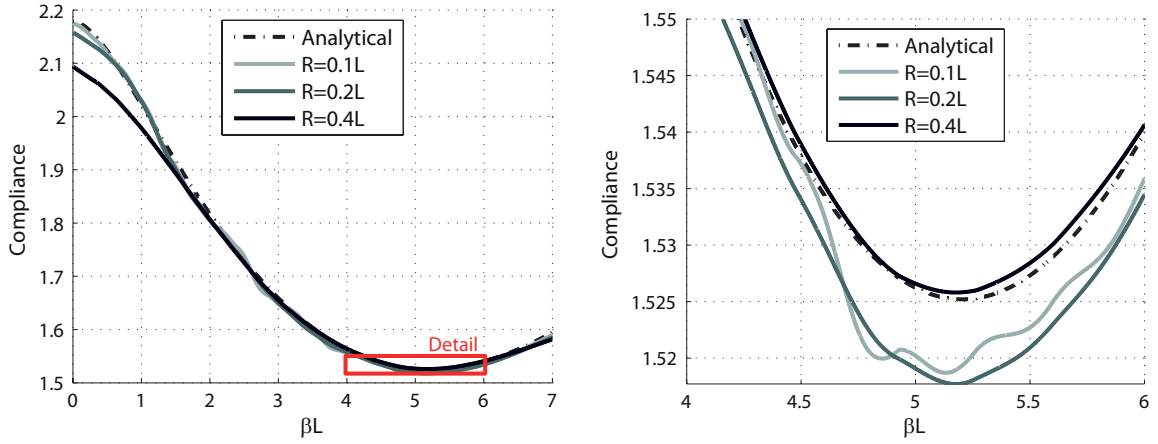


(a) (b)

Figure 2.9: (a) Gradient with convolution coupling for different mesh refinements. (b) Detail close to the optimum.

Table 2.1: One-dimensional bar with cable: optimal anchor location for randomly generated discretizations with different levels of refinement.

	β_c	$C(\beta_c L)$
Exact	0.7434	1.5252
$N_E = 10$	0.7492	1.5144
$N_E = 20$	0.7346	1.5169
$N_E = 40$	0.7431	1.5203



(a) (b)

Figure 2.10: (a) Compliance with convolution coupling for different convolution radiuses. (b) Detail close to the optimum.

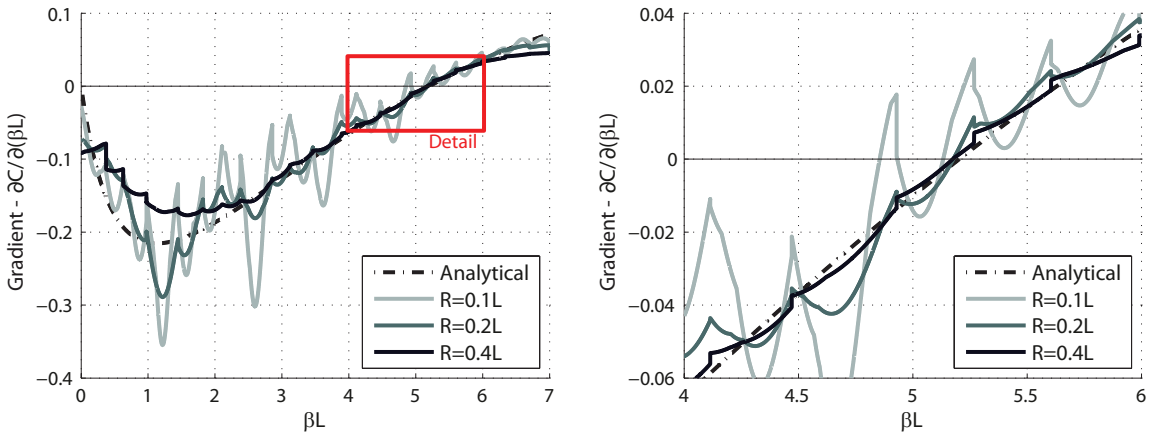


Figure 2.11: (a) Gradient with convolution coupling for different convolution radius. (b) Detail close to the optimum.

Table 2.2: One-dimensional bar with cable: optimal anchor location with varying convolution radius for a randomly generated discretization with $N_E = 20$.

	β_c	$C(\beta_c L)$
Exact	0.7434	1.5252
$R = 0.1L$	0.7333	1.5187
$R = 0.2L$	0.7396	1.5177
$R = 0.4L$	0.7393	1.5258

with the FEM-based and convolution-based couplings. The embedding using FEM shape functions suffers from discontinuities at the element boundary and $\partial C/\partial x = 0$ at several points, thus is prone to converge at the many local minima, far from the global optimum. Convolution coupling is continuous, and inspection of the gradient indicate that it is likely to converge close to the actual (analytical) optimum.

The optimization problem for 50 iterations, with a starting point $\beta_0 = 0.5$ is performed for $N_E = 20$ (element mesh), with randomly spaced elements of size $0.7L/N_E \leq \Delta x \leq 1.3L/N_E$. The only constraint or technique used is the move limit as detailed in Equation (2.23) with $m = 0.1$. The optimizer is the *Method of Moving Asymptotes* (MMA) (Svanberg, 1987). The convergence towards the optimal point $\beta_c L$ is shown in Figure 2.13(a) and the compliance plot in Figure 2.13(b). There is an oscillatory behavior between iterations 17 and 30 due

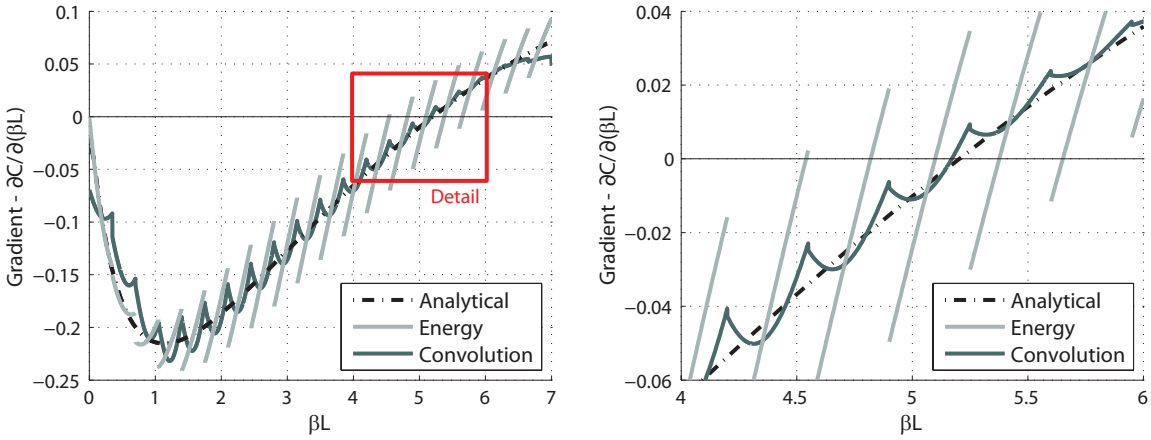


Figure 2.12: (a) Sensitivity plot for analytical, FEM-based and convolution-based shape functions. (b) Detail close to the optimum.

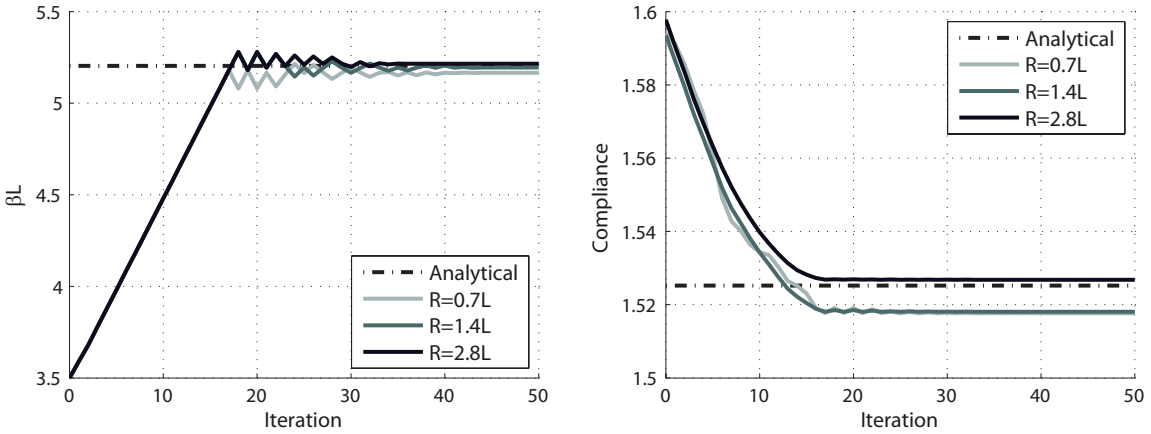


Figure 2.13: Optimization evolution for 50 iterations with different convolution radiuses. (a) Anchor point βL. (b) Compliance.

to the *adventurous* behavior of the optimizer close to the optimum. The oscillations can be eliminated by taking a smaller move limit, or decreasing it with each iteration.

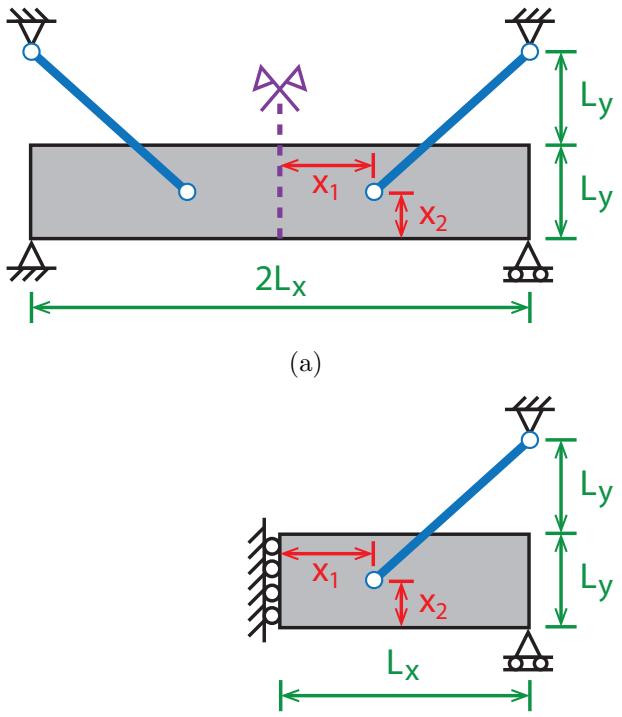


Figure 2.14: Deep beam with cable supports subjected to self-weight. (a) Idealized model. (b) Model considering the symmetry of the problem.

2.2.2 Deep beam with cable anchors

This problem, introduced in Figure 2.1, tries to find the optimal anchoring location of two (symmetric) cables on a simply supported deep beam loaded by self-weight. The problem is modeled as in Figure 2.14(a): using symmetry the problem is reduced to finding the optimal position of a single cable (constant area) on a half domain as in Figure 2.14(b). The half domain has size $L_x \times L_y$ and is loaded by self-weight b, the domain is regularly partitioned in $N_x \times N_y$ four node quadrilateral elements (Q4). The convolution function being used is $h_2(r)$ presented in Equation (2.17). The design variables of the problem are the anchor location coordinates x_1 and x_2, with the only constraint or technique being the move limit as in Equation (2.23) with $m = 0.05$.

The problem parameters are $L_x = 2$ ($2L_x = 4$), $L_y = 0.8$, $b = -2$, $E_c = 100$, $\nu = 0.3$ and

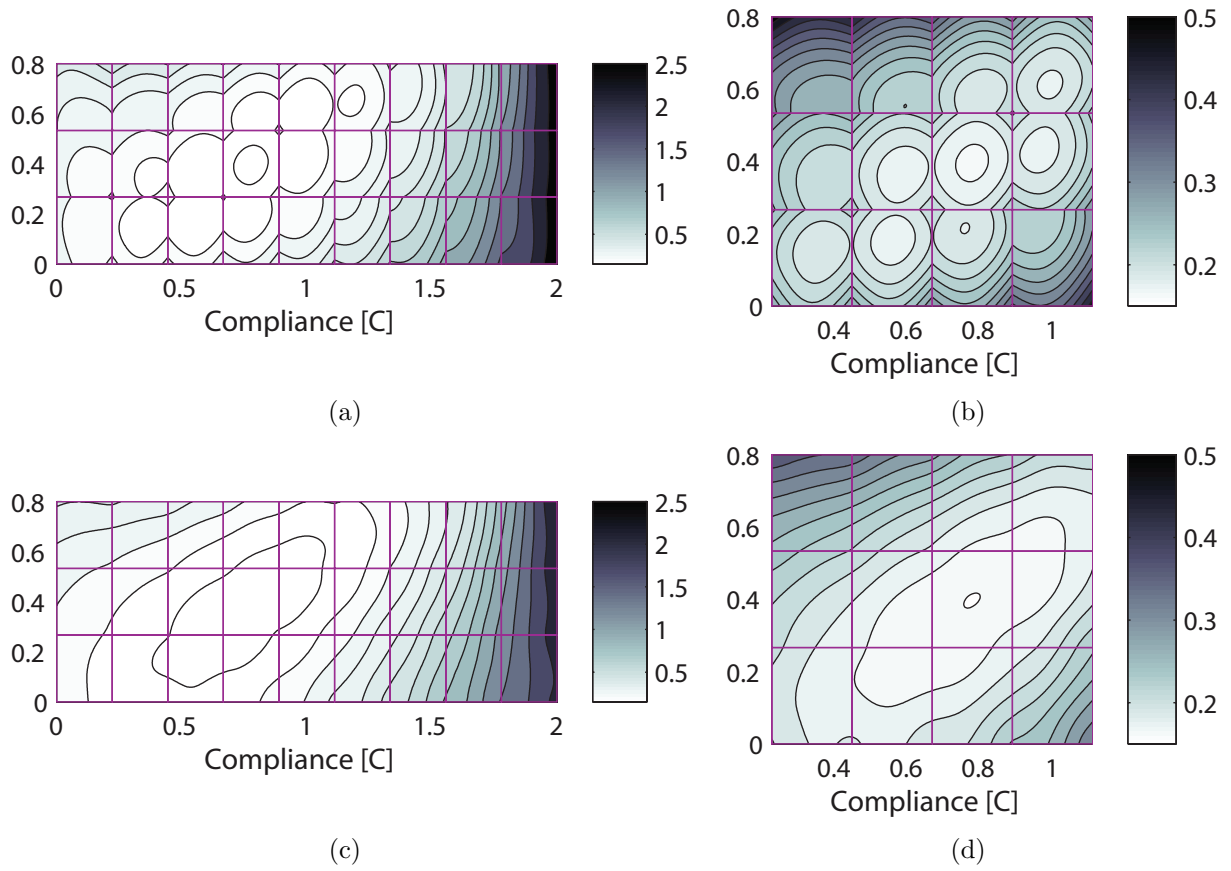


Figure 2.15: Objective function plot for the deep beam with cable support problem using a 9×3 Q4 element mesh for the continuum. (a) FEM-based coupling. (b) Detail of FEM-based coupling near the global optimum. (c) Convolution-based coupling with $R = 0.5$. (d) Detail of convolution-based coupling with $R = 0.5$.

$EA = 300$. The objective function (compliance) using FEM-based and convolution-based coupling for an $N_x = 9$ and $N_y = 3$ mesh are plotted in Figure 2.15. The convolution-based coupling exhibits what seems to be a single unique optimum, whereas the FEM-based coupling has a tendency to produce a minimum at the continuum element's centroid.

Refining the continuum with $N_x = 20$ and $N_y = 8$, the differences in the objective function (compliance) plots become more apparent. The FEM-based coupling is now plagued by multiple local minima (Figures 2.16(a) and 2.16(b)), whereas the convolution-based coupling became smoother due to the larger number of points within the convolution operator (Figures 2.16(c) and 2.16(d)). With this finer mesh, we can afford to reduce the radius

Table 2.3: Deep beam with cable anchors: optimal anchor location and compliance for an increasingly refined Q4 mesh and $R = 0.3$.

Mesh	x_1	x_2	$C(x_1, x_2)$
10×04	0.9999	0.5100	0.1956
20×08	0.8541	0.4016	0.1910
40×16	0.8094	0.3712	0.1971
80×32	0.8635	0.3711	0.2025

Table 2.4: Deep beam with cable anchors: optimal anchor location and compliance for an increasingly refined Q9 mesh and $R = 0.3$.

Mesh	x_1	x_2	$C(x_1, x_2)$
10×04	0.8497	0.4030	0.1952
20×08	0.8688	0.3736	0.2006
40×16	0.9004	0.3833	0.2056
80×32	0.9272	0.3721	0.2101

of the convolution operator while still preserving a smooth and continuous field (Figures 2.16(e) and 2.16(f)). This reduction is desirable when possible to minimize the error that is being introduced. The convolution-based coupling with $R = 0.3$ gives $x_1 = 0.8165$ and $x_2 = 0.3699$ as the global optimum for a 20×8 mesh. However, the global optimum location does change with the mesh refinement, and the solution experiences small changes when the mesh is changed.

The problem is optimized for 30 iterations with a starting point $[x_1, x_2] = [L_x, L_y/2]$ measured from the bottom left corner of the half-domain, using a convolution radius of $R = 0.3$. The optimizer is again the *Method of Moving Asymptotes* (MMA) (Svanberg, 1987). Mesh convergence results are available in Table 2.3 and Figure 2.17.

The method makes no distinction of the element type or mesh. The problem is re-meshed with 9 node quadrilateral elements (Q9) and tested with different mesh refinements and convolution radii. The evolution of the objective function throughout the iterations for both methods are plotted in Figures 2.18(a) and 2.18(b), and in both situations a smooth decrease is observed. The final converged results, available in Tables 2.4 and 2.5, reinforce the fact that the method is relatively stable: variations can be seen again with refinement, but these all oscillate about the same general area.

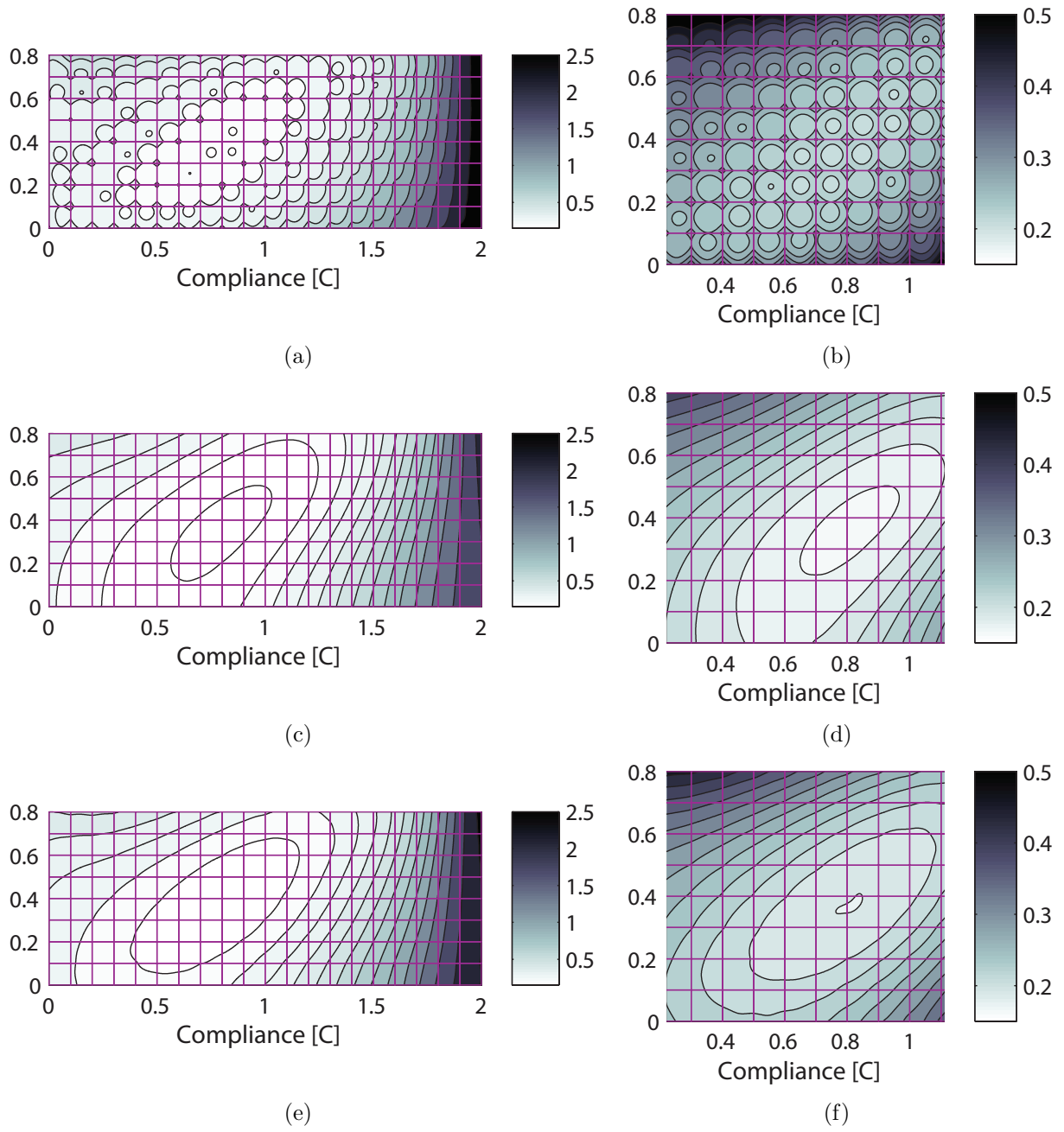


Figure 2.16: Objective function plot for the deep beam with cable support problem using a 20×8 Q4 element mesh for the continuum. (a) FEM-based coupling. (b) Detail of FEM-based coupling near the global optimum. (c) Convolution-based coupling with $R = 0.5$. (d) Detail of convolution-based coupling with $R = 0.5$. (e) Convolution-based coupling with $R = 0.3$. (f) Detail of convolution-based coupling with $R = 0.3$.

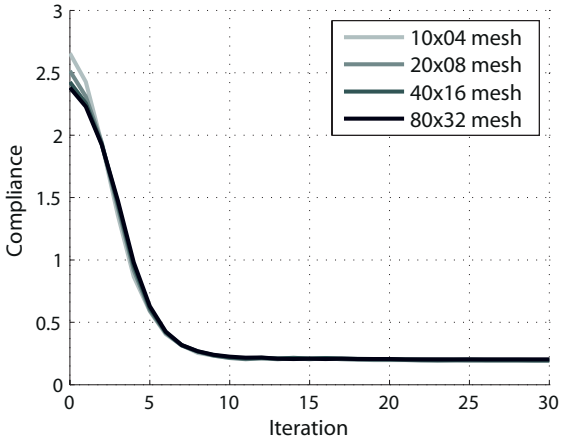
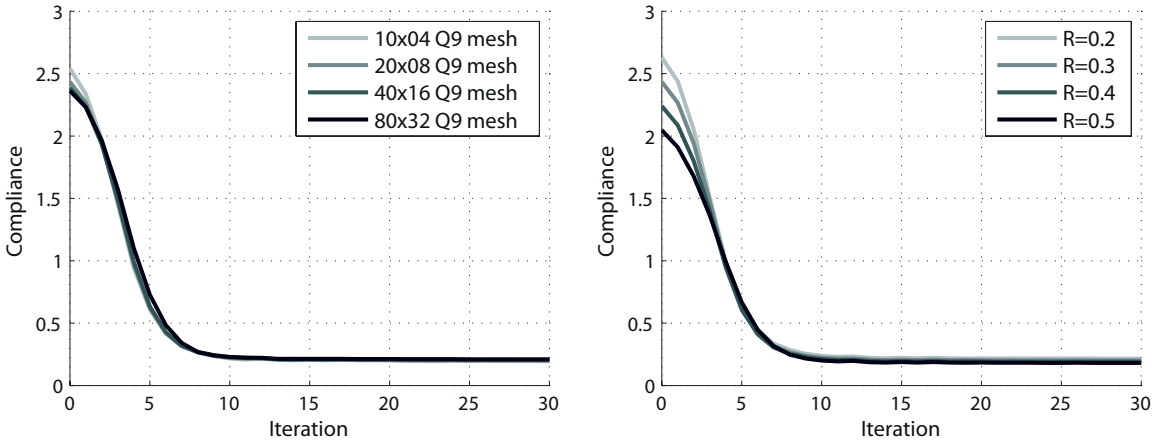


Figure 2.17: Evolution of the compliance for the beam with cable anchors problem. Optimization was done with 30 iterations, $R = 0.3$, and using increasingly refined Q4 meshes.



(a) (b)

Figure 2.18: Optimization for beam with cable anchor using Q9 elements. (a) Compliance evolution for increasingly refined meshes and $R = 0.3$. (b) Compliance evolution for a 20×8 mesh with varying radius.

Table 2.5: Deep beam with cable anchors: optimal anchor location and compliance for a 20×8 Q9 mesh with varying convolution radius.

	x_1	x_2	$C(x_1, x_2)$
$R = 0.2L$	0.8763	0.4025	0.2154
$R = 0.3L$	0.8688	0.3736	0.2006
$R = 0.4L$	0.8747	0.3752	0.1900
$R = 0.5L$	0.8769	0.3748	0.1816

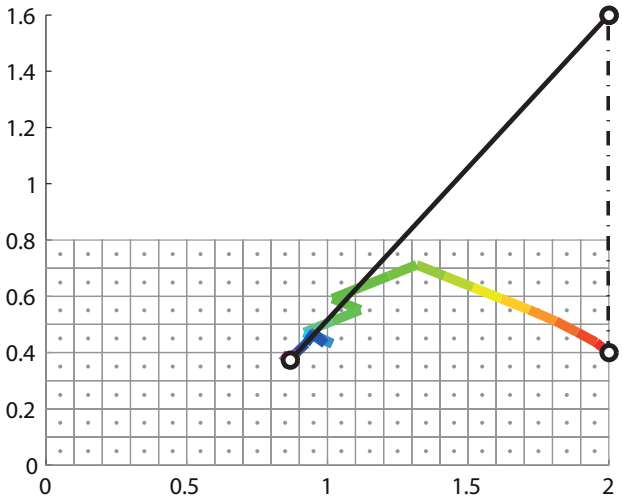


Figure 2.19: Deep beam with cable anchors: optimization for a 20×8 Q9 element mesh showing the anchor path throughout the iterations.

The anchor path throughout the iterations for this problem is shown in Figure 2.19. This path exhibits a steady and consistent approach towards the optimal solution, where the cable most efficiently supports the continuum.

2.3 Examples

The examples explored here showcase some of the problems this method can address. The optimizer is the *Method of Moving Asymptotes* (MMA) (Svanberg, 1987), and the convolution function used is $h_2(\cdot)$ from Equation (2.17). Problems are optimized for minimum compliance ($J = \mathbf{u}^T \mathbf{K}_e \mathbf{u}$) of the coupled structure. For the specific case of two-dimensional problems, unit thickness and plane stress is assumed.

2.3.1 Tapered building with truss superstructure

This problem explores extending the method to a larger number of elements (and design variables). All truss members are embedded within the continuum. In addition, the continuum is modeled with an unstructured mesh. A sketch of the continuum domain, with a

Table 2.6: Building with truss superstructure: Final nodal locations for the symmetry constrained and free problems with node numbering in accordance with Figure 2.20(b).

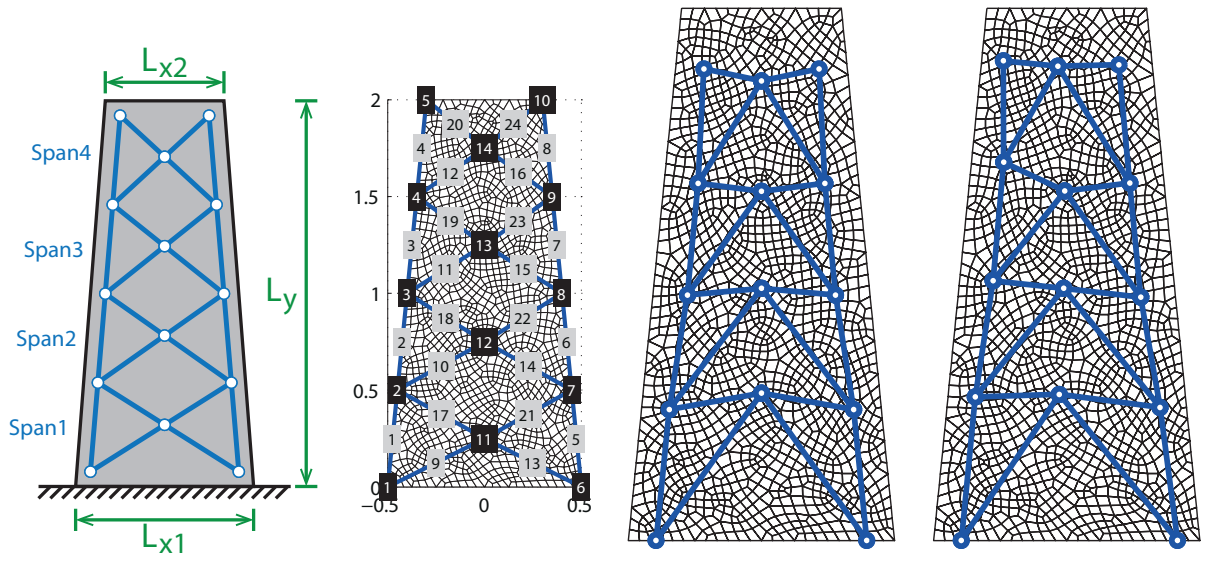
	Symm	Free		Symm	Free
x_1	-0.3958	-0.3959	y_1	0.0000	0.0000
x_2	-0.3522	-0.3433	y_2	0.5376	0.5406
x_3	-0.2770	-0.2733	y_3	0.9426	0.9546
x_4	-0.2379	-0.2491	y_4	1.3449	1.4098
x_5	-0.2188	-0.2385	y_5	1.7725	1.8019
x_6	0.3958	0.4167	y_6	0.0000	0.0000
x_7	0.3522	0.3547	y_7	0.5376	0.5027
x_8	0.2770	0.3063	y_8	0.9426	0.9485
x_9	0.2379	0.2275	y_9	1.3449	1.3711
x_{10}	0.2188	0.2086	y_{10}	1.7725	1.8003
x_{11}	0.0000	-0.0285	y_{11}	0.5544	0.5494
x_{12}	0.0000	0.0535	y_{12}	0.9420	0.9122
x_{13}	0.0000	-0.0201	y_{13}	1.3124	1.3115
x_{14}	0.0000	-0.0318	y_{14}	1.7901	1.7828

truss superstructure is shown in Figure 2.20(a), where the truss superstructure links to the continuum at the node locations. The problem is optimized with 4 spans, and a starting position as shown in Figure 2.20(b), with the nodes numbered as in the Figure.

The continuum is meshed with $N_E = 1520$ Q8 elements, with dimensions and material properties: $L_{x1} = 1.0$, $L_{x2} = 0.6$, $L_y = 2$, $E_c = 10$, $\nu = 0.3$. The truss consists of 4 spans with equal properties for all bars $EA = 300$ and convolution radius $R = 0.075$. The structure is loaded by self-weight of the continuum $b = -10$. The design variables are the nodal positions of the truss (cross-sectional areas are not being optimized). The problem is optimized for 50 iterations, with a coordinate move limit of $m = 0.015$ as in Equation (2.23), and a truss volume constraint of $V_{\max} = 32$ (note that initially the truss has a volume $V_0 = 34.76$) in accordance with Equation (2.27).

The optimization is performed for the case where symmetry is imposed, and for when is not. The final configurations for both cases can be seen in Figures 2.20(c) and 2.20(d), and the final nodal locations are reported in Table 2.6. The unsymmetrical mesh in the continuum causes the truss to loose symmetry, and it is unable to recover.

The compliance plot in Figure 2.21(a) has an initial increase while the optimizer is fulfill-



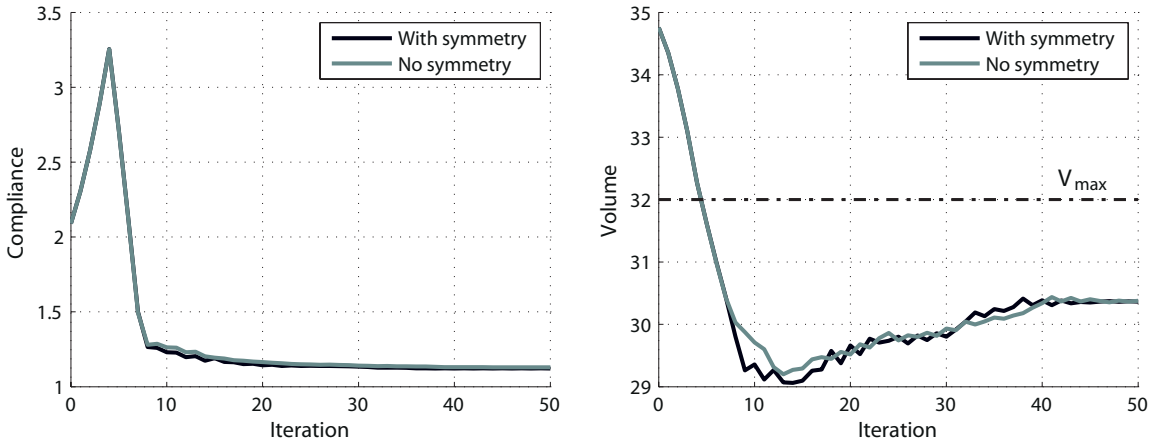
(a) (b) (c) (d)

Figure 2.20: Building with truss superstructure. (a) Domain and truss specifications. (b) Starting configuration with node and element numbering with 4 spans, $L_{x1} = 1.0$, $L_{x2} = 0.6$ and $L_y = 2$. (c) Final configuration with symmetry along the mid vertical axis imposed. (d) Final configuration with symmetry not imposed.

ing the truss volume constraint, as shown in Figure 2.21(b). Once the constraint is satisfied, the optimizer is free to search for the optimal truss geometry (using the node locations only). The final compliance for the symmetry imposed and free cases are $C_{symm} = 1.1215$ and $C_{free} = 1.1296$. The optimized compliance for the symmetric case is surprisingly lower. However, if iterations continue, the less-constrained unsymmetric case will have a lower final value. The unsymmetric case has more than twice the number of design variables compared to the symmetric case, resulting in a (slightly) lower rate of convergence.

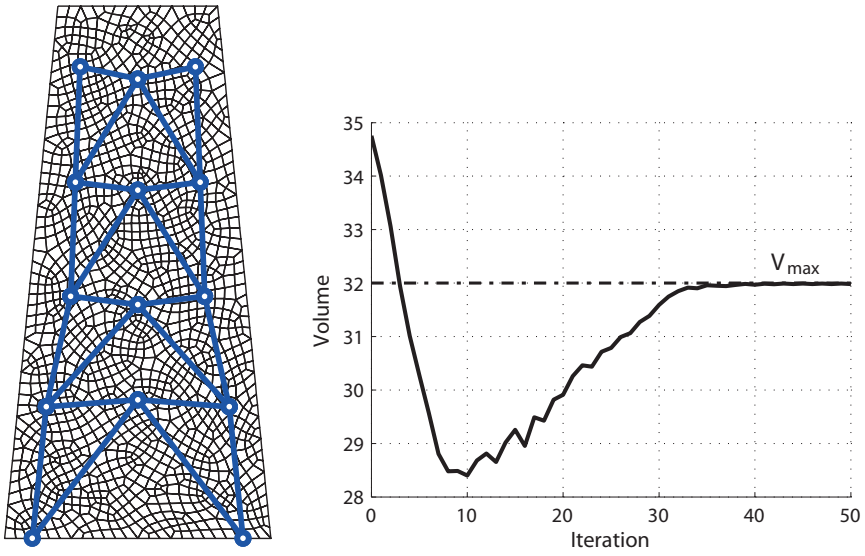
2.3.2 Full truss layout optimization for tapered building

This is an extension of the previous problem, adding the truss member's cross-sectional areas as design variables for the optimization of the symmetric case. The simultaneous optimization of member sizing and geometry translates into a full layout optimization of the building's truss superstructure. Previously, the final volume of the truss did not match $V_{\max}$



(a) (b)

Figure 2.21: Optimization for building with truss superstructure (design variables are nodal coordinates) for 50 iterations. (a) Compliance evolution throughout the optimization. (b) Volume evolution throughout the optimization.



(a) (b)

Figure 2.22: Full layout optimization of the building's truss superstructure (design variables are nodal coordinates and member cross-sectional areas) for 50 iterations. (a) Final geometry. (b) Volume evolution throughout the optimization.

because the design variables were the node locations only (Figure 2.21(b)). The gradient of the cross-sectional areas follow Equation (2.26). The constraints in Equation (2.27) are also used with $A_{\min} = 0.015$ and $m_a = 0.015$.

Table 2.7: Building with truss superstructure (full layout optimization): Final cross-sectional areas for truss members in accordance with Figure 2.20(b).

A_1	3.6269	A_9	3.1658	A_{17}	3.0460
A_2	3.6165	A_{10}	3.1296	A_{18}	2.7400
A_3	3.5832	A_{11}	3.0016	A_{19}	2.7346
A_4	3.3270	A_{12}	2.8983	A_{20}	2.6999
A_5	3.6269	A_{13}	3.1658	A_{21}	3.0460
A_6	3.6165	A_{14}	3.1296	A_{22}	2.7400
A_7	3.5832	A_{15}	3.0016	A_{23}	2.7346
A_8	3.3270	A_{16}	2.8983	A_{24}	2.6999

The optimized element areas following the element numbering scheme from Figure 2.20(b) are detailed in Table 2.7. The node locations exhibit minimal variation with respect to the previous symmetric case (Figure 2.22(a)). As expected, the optimizer allocated the previously unused volume, further improving the solution and resulting in an active volume constraint (Figure 2.22(b)). The final compliance after 50 iterations is equal to $C_{full} = 1.0977$. This value is lower than in the previous cases, as expected, because of the larger volume of the truss in the final structure.

2.3.3 Three-dimensional beam with truss reinforcements

This problem explores the optimal position of a reinforcing truss within a three-dimensional beam. Because the truss only links with the continuum at the nodes, the bars have no connection along their length with the continuum: this can be interpreted as if the truss members are within a casing allowing them to slide between nodes. The domain definition and initial bar location is given in Figure 2.23(a). The design variables are the node locations, that are initially positioned as specified in Table 2.8, following the node numbering from Figure 2.23(a). The domain is meshed with Tet10 elements dividing the domain in $N_x \times N_y \times N_z = 36 \times 11 \times 8$ blocks, with each block subdivided in 6 Tet10 elements for a total of $N_E = 19008$ Tet10 elements (Figure 2.23(b) is a deformed plot of the mesh). In addition, the dimensions are $L_x = 10$, $L_y = 3$, $L_z = 2$, and the material properties for the continuum are $E_c = 100$ and $\nu = 1/3$. All truss members have equal properties; $EA = 500$. The

Table 2.8: Initial truss nodal locations within the three-dimensional beam.

Node	x	y	z
1	0.5000	1.2000	1.6000
2	3.0000	1.2000	0.4000
3	7.0000	1.2000	0.4000
4	9.5000	1.2000	1.6000
5	0.5000	1.8000	1.6000
6	3.0000	1.8000	0.4000
7	7.0000	1.8000	0.4000
8	9.5000	1.8000	1.6000

Table 2.9: Final truss nodal locations within the three-dimensional beam.

Node	x	y	z
1	0.4972	0.6366	1.2482
2	2.2785	0.7085	0.0000
3	7.7148	0.7021	0.0000
4	9.5002	0.6281	1.2542
5	0.5005	2.3713	1.2562
6	2.2828	2.2935	0.0000
7	7.7181	2.2898	0.0000
8	9.5012	2.3635	1.2485

only constraint or restriction included is a move limit in the nodal coordinates $m = 0.1$, in accordance with Equation (2.23). The beam is loaded by a distributed load acting on the top face $b = -2$, and the problem is optimized for compliance for 30 iterations with a convolution radius $R = 0.5$

The problem does not have symmetry imposed, and the final nodal coordinates after 30 iterations are in Table 2.9. Nevertheless, within some numerical precision, symmetry is preserved. The final compliance for the problem is $C = 79.5418$, and the evolution throughout the iterations is presented in Figure 2.24, again with a smooth decrease towards the optimum.

2.3.4 Reinforced double corbel

This example deals with the steel layout of a double corbel based on Example 3.2 in the ACI SP-208 (ACI Committee, 2002). The corbel transfers beam reaction forces $V_u = 61.8\,kips$

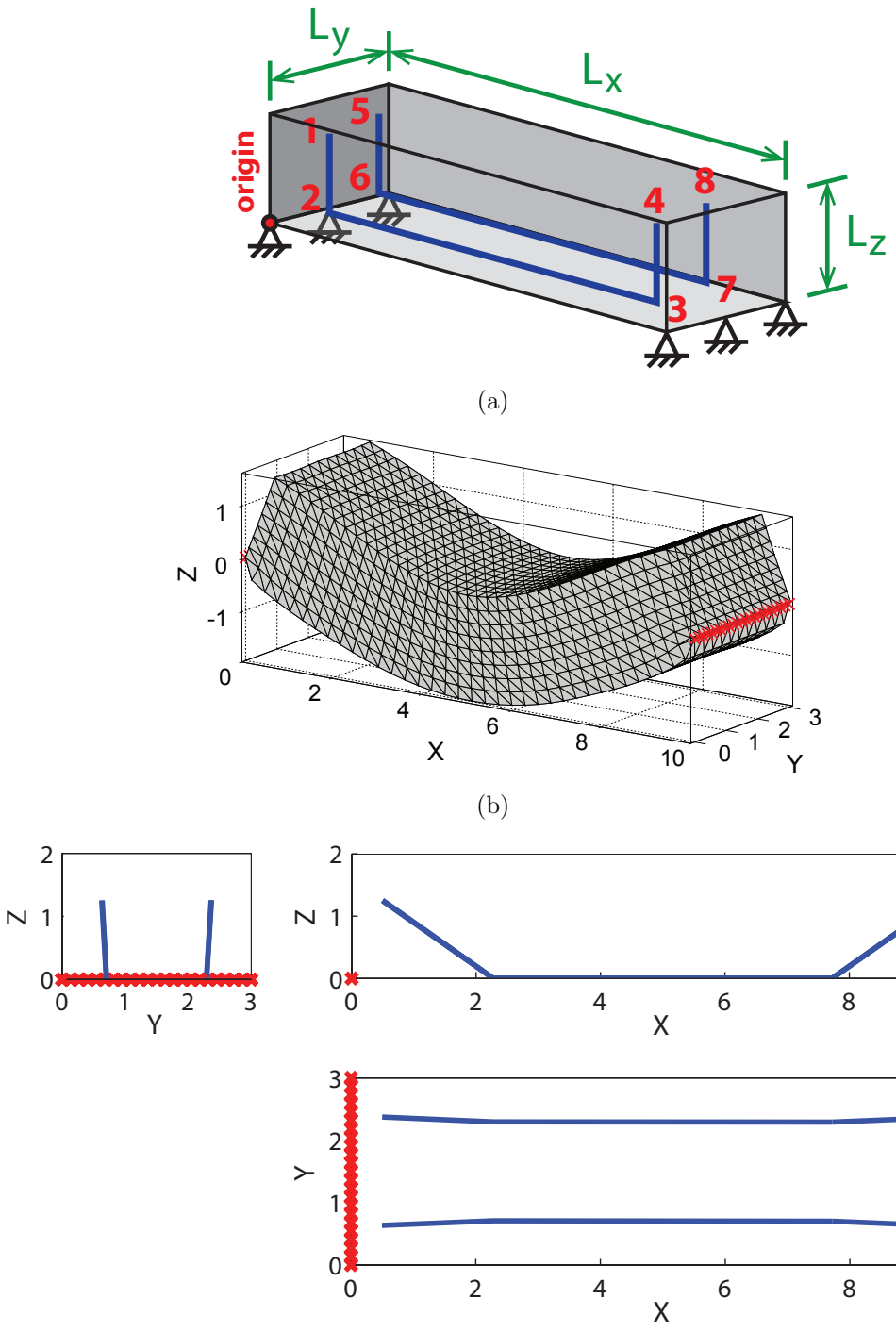


Figure 2.23: Optimization for a three-dimensional beam with an embedded truss. (a) Domain definition and node numbering. (b) Continuum meshed with Tet10 elements in the final deformed state. (c) Front, side and top views of the final configuration.

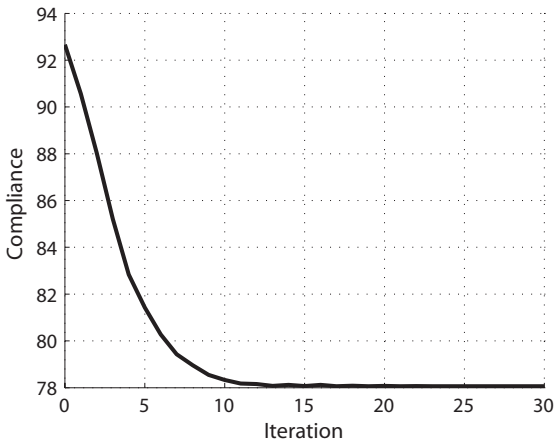


Figure 2.24: Three-dimensional beam with an embedded truss: compliance evolution for 30 iterations.

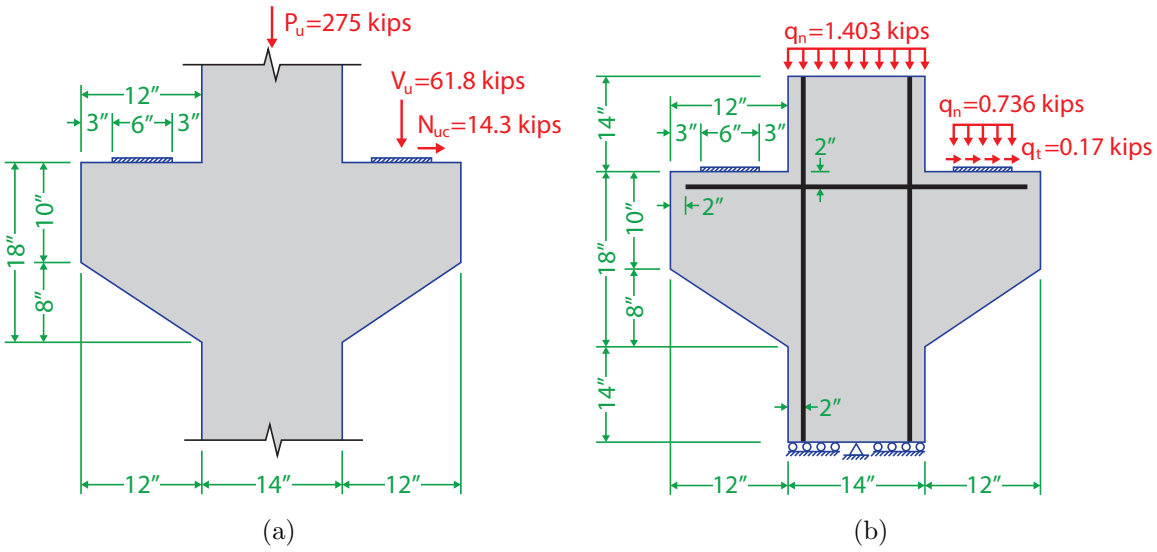


Figure 2.25: Double corbel problem definition. (a) Problem definition in accordance with ACI SP-208. (b) Model domain, loads and boundary conditions.

and $N_{uc} = 14.3\,kips$ to a square 14 in column through a 6 in plate as depicted in Figure 2.25(a). In addition, the upper column carries a compressive axial load $P_u = 275\,kips$. The problem deals with the layout of the steel in tension, initially placed 2 in below the corbel supports.

The loads coming from the upper column and the beams are distributed over the column

cross-sectional area and plate respectively. Analysis for the continuum will be carried out on a $t = 1\,in$ thick model, with *plane stress*. Given the depth dimension of the corbel, a three-dimensional analysis would be more appropriate, but the simplicity of a *plane stress* analysis is more appropriate to showcase the method in an application setting. The vertical steel in compression has a cross-sectional area $A_{sc} = 0.1\,in^2$ (not to be designed), and the steel in tension has initially $A_{st} = 0.1\,in^2$. The elastic modulus of steel is $E_s = 29000\,kips$, and for the concrete $E_c = 3600\,kips$ and $\nu = 0.2$. The model with the loads, boundary conditions and initial steel placement (for a $1\,in$ thick model) is presented in Figure 2.25(b).

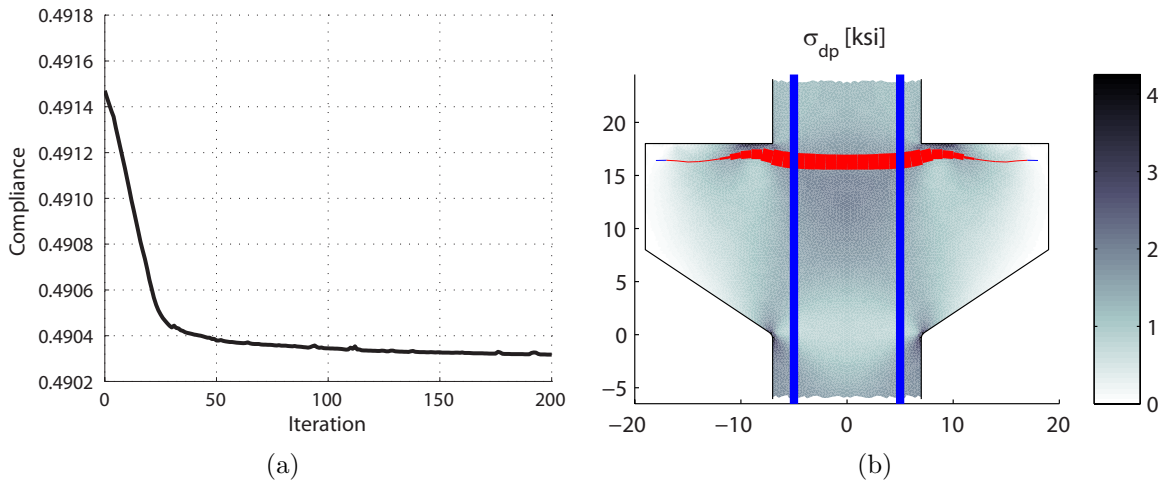
The concrete is modeled using 23312 T6 elements, and 47065 nodes. The steel rebars are modeled as several pin-jointed bars $1\,in$ apart to allow for linkage with the continuum throughout the length of the bar. The convolution radius is $R = 0.25\,in$. The optimization is done for compliance subject to constant volume, and the design variables are steel cross-sectional areas of the bars, and the vertical (y direction) node positions of the bar in tension (layout optimization). The node movement is limited to $1\,in$ away from the concrete edges to allow for steel cover. The constraints or restrictions included are a move limit $m = 0.1$ as in Equation (2.23) for the node locations, and in the cross-sectional areas $m_a = 0.005\,in^2$ and $A_{\min} = 0.001\,in^2$ as in Equation (2.27). The optimization is run for 200 iterations for a symmetric mesh, with symmetry not enforced.

Experimental results (Imran and Pantazopoulou, 1996) suggest the following Drucker-Prager model for concrete:

$$0.3\left(\frac{I_1}{f'_c} + 1\right) = \sqrt{\frac{1}{3}} - \frac{\sqrt{J_2}}{f'_c}\,, \tag{2.33}$$

where I_1 and J_2 are the first and second principal stress invariant. Reorganizing the terms:

$$\frac{0.27 + 0.3\sqrt{3}}{0.73} I_1 + \frac{1 + 0.3\sqrt{3}}{0.73}\sqrt{3J_2} = f'_c\,, \tag{2.34}$$



(a) (b)

Figure 2.26: Double corbel optimization results after 200 iterations. (a) Compliance evolution throughout the iterations. (b) Final steel layout and concrete Drucker-Praguer stress.

we can define the Drucker-Prager stress as:

$$\sigma_{dp} = 1.0817 I_1 + 2.0817\sqrt{3J_2} \tag{2.35}$$

Failure occurs when $\sigma_{dp} = f'_c$, analogous to Von Mises stress. The concrete used in the documented example in (ACI Committee, 2002) is assumed to have $f'_c = 4\, ksi$.

The compliance plot (Figure 2.26(a)) exhibits a smooth decrease throughout the iterations, but with little improvement after the optimization. Despite not being enforced, symmetry was indeed preserved as expected. The final position of the steel and the cross-sectional areas are in Figure 2.26(b) (blue and red color indicate steel in compression and tension respectively), as well as a Drucker-Prager stress.

The gain is clear if the steel is looked at in detail: the bar orients itself towards the principal directions taking a *moustache shape*. The optimized cross-sectional areas vary as in Figure 2.27(a), but most importantly, the bar takes a constant stress throughout its length as in Figure 2.27(b), in accordance with Michell's fully stressed requirements (Hemp, 1973; Michell, 1904; Rozvany, 1996, 1997b). In the final configuration there is no shear in the bar,

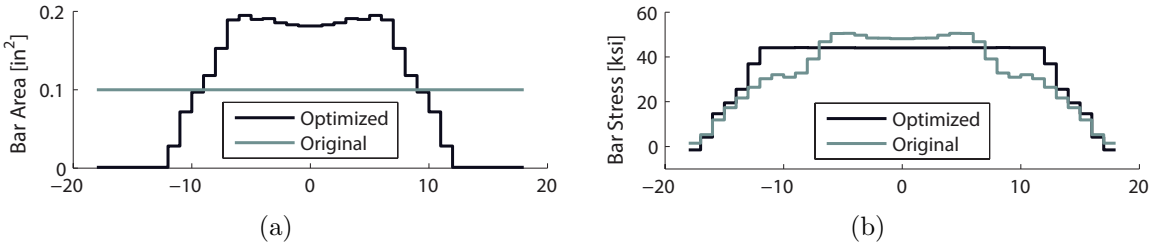


Figure 2.27: Double corbel optimized steel in tension. (a) Cross-sectional area. (b) Axial stress.

that, along with the constant stress (smaller than the previous maximum stress), makes a more efficient use of the steel available and thus a better design. The final position and cross-sectional areas for the (half) bar are in Tables 2.10 and 2.11 respectively.

Table 2.10: Double corbel optimization: final node locations for one symmetric half of the steel in tension (in).

Node	x	y	Node	x	y
1	0	16.2521	11	10	16.9086
2	1	16.2555	12	11	16.7158
3	2	16.2695	13	12	16.5538
4	3	16.2921	14	13	16.4070
5	4	16.3344	15	14	16.2939
6	5	16.3995	16	15	16.2710
7	6	16.5203	17	16	16.3699
8	7	16.7265	18	17	16.4352
9	8	16.9813	19	18	16.4119
10	9	17.0000			

Table 2.11: Final cross-sectional areas for one symmetric half of the steel in tension. Values given for segments between nodes i and j (in^2).

$node_i$	$node_j$	A_s	$node_i$	$node_j$	A_s
1	2	2.5389	10	11	1.3553
2	3	2.5596	11	12	1.0042
3	4	2.5937	12	13	0.3947
4	5	2.6694	13	14	0.0148
5	6	2.6540	14	15	0.0144
6	7	2.7265	15	16	0.0143
7	8	2.6469	16	17	0.0143
8	9	2.1364	17	18	0.0142
9	10	1.6514	18	19	0.0142

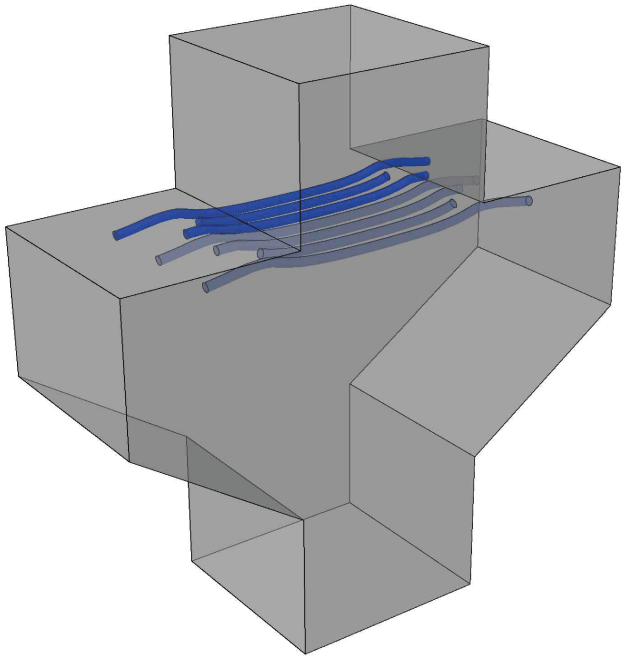


Figure 2.28: Double corbel with optimized steel in tension.

The steel for the whole 14 in thick corbel is laid in one layer and 3 different lengths following the results from the optimization as in Table 2.12. The corbel with the optimized steel design is presented in Figure 2.28. This problem only optimizes and designs the primary reinforcement; Additional shear reinforcement and hooks are required for the design to be treated seriously.

Table 2.12: Corbel reinforcement steel in traction.

Rebar	Horizontal Position*
3#5	$-12.5\,in$ to $12.5\,in$
2#5	$-9.0\,in$ to $9.0\,in$
2#5	$-7.5\,in$ to $7.5\,in$
$\star$ Lengths measured horizontally	

2.4 Conclusions

The method presented here extends truss layout optimization to the situation when it is embedded within a continuum, allowing for mixed-element type optimization problems to

be solved. The method is based on a convolution operator to link the truss node to the continuum. The derivative field remains continuous and sufficiently smooth, and gradient-based optimizers can be used, even for small convolution radii.

Convolution coupling to the continuum does violate the energy principle of the problem, but when used with a reasonable sized convolution radius, the results are shown to agree up to some degree with an exact solution when available. In cases where an analytical solution cannot be easily found, the method exhibits stable results (i.e. converging to similar solutions for a wide range of initial conditions). It is expected for the optimum solution to have small variations, that are attributed to the difference in the FEM solutions with refinement, and numerical inaccuracies.

There is no restriction over the objective function in the optimization, provided that the derivation for the stiffness term follows Equation (2.21). Restrictions to the optimization are easily implemented and examples with volume constraint, minimum cross-sectional areas and member lengths are given. The method does require a move limit between iterations due to the highly nonlinear behavior of the problems involving geometric optimization. The situation worsens with an increased number of truss nodes or the inclusion of member sizing, and thus the optimization can easily diverge.

The method is shown to effectively reach optimal configurations. However, an *acceptable* initial guess must be given, because of the large number of local minima in these problems. Note that a truss can have an infinite number of spatial configurations, thus, relying on the *engineer's common sense* to provide a starting point for the optimization.

Chapter 3

Lateral bracing systems in 2D and 3D

Structural optimization has a long history of applications with buildings. Lateral bracing systems are often used to provide lateral stiffness to buildings. These may span one or several bays, single or several stories high (Figure 3.1).

Density–based topology optimization has been applied to optimal bracing system problems (Neves et al., 1995; Mijar et al., 1998; Allahdadian et al., 2012; Stromberg et al., 2012), with the finding that the optimal bracing point is not always at mid–height. Density–based topology optimization is a powerful technique, but the interpretation of the results and subsequent member sizing is not straightforward. An alternative approach is the *ground structure* method (Dorn et al., 1964; Ben-Tal and Bendsøe, 1993; Sokół, 2011). The ground structure method results in solutions with a large number of members, that asymptotically converge to the theoretical optimum for problems with known (analytical) solutions (Michell, 1904; Hemp, 1973). The approach used in this work simultaneously optimizes truss geometry (node locations) and member sizes (Felix and Vanderplaats, 1987; Hansen and Vanderplaats, 1990; Lipson and Gwin, 1977), as is often called *truss layout optimization*. The structural connectivity is fixed, so no members are added or removed. Zero cross–sectional areas are a special case which cannot be allowed due to a known discontinuity in member's stresses (Sved and Ginos, 1968; Kirsch, 1990; Rozvany, 2001). However, in most applications, a very small cross–sectional area has an effect similar to removing the bar. The bracing system is

(a) (b) (c)

Figure 3.1: Examples of single and multiple bays braced buildings. (a) The John Hancock Center — Chicago, Illinois, USA [SOM | Ezra Stoller © Esto]. (b) The Alcoa Building — San Francisco, California, USA [SOM | © Mak Takahashi]. (c) Building in Presidente Riesco Ave, Santiago, Chile [© Tomás Zegard].

modeled as an elastic truss with static loads and small displacements. The connection costs are considered to be constant or null. This means that the cost to connect members with different cross–sectional areas and angles is assumed not to change.

3.1 Four complementary formulations

The question of where is the optimal bracing point, is actually a subjective one: the math and physics involved are exact, but to define an optimum, a measure or benchmark must be chosen. The options for the objective function (measure) are limitless, but only a handful are of interest to the practicing engineer. The following four objectives are explored in this work:

1. Minimize the volume

2. Minimize the load–path

3. Minimize compliance

4. Minimize displacements

These objectives require constraints in order for the solution to be bounded and unique. In all cases, the structural internal–external force equilibrium is enforced, either by $\mathbf{Ku} = \mathbf{f}$, or an equivalent expression. The problems considered in this paper are elastic lateral bracing systems with small deformations, and no self–weight or connection costs (or constant). Nonetheless, some of the concepts and conclusions can be extended to a wider range of options and constraints (e.g. buckling or frequencies). Each formulation has some properties, advantages and disadvantages. A brief discussion of these will be presented in the following section.

3.1.1 Volume formulation

An intuitive formulation for a practicing engineer is to minimize the volume of structural material. Typically, the cost of a structure is proportional to its weight. Thus, minimizing the total weight of the structure minimizes its cost (when the connection and joint costs are constant). A stress constraint is required as it prevents the member's cross–sectional areas from approaching zero. The *minimum volume formulation* is as follows (with *s.t.* standing for *subject to*):

$$\begin{aligned} \min_{\mathbf{A},\mathbf{x}} \quad & V = \mathbf{A}^T\mathbf{L} \\ \text{s.t.} \quad & \sigma_c \leq \sigma_i \leq \sigma_t \qquad \forall i = 1 \dots n_e \\ & \mathbf{A} \geq \mathbf{0} \\ \text{with} \quad & \mathbf{Ku} = \mathbf{f}\,, \end{aligned} \tag{3.1}$$

where $\mathbf{L}$ and $\mathbf{A}$ are column vectors with the member lengths and cross–sectional areas respectively, $\mathbf{x}$ is a vector with the joint coordinates, $\mathbf{K}$ the stiffness matrix, $\mathbf{u}$ the nodal displacements, $\mathbf{f}$ the nodal force vector, σ_i the member stresses and σ_c and σ_t the stress limits on compression and tension respectively.

The volume formulation is arguably the most intuitive and common (Michell, 1904; Hemp, 1973). The strength of this formulation is that dealing with different stress limits for compression and tension, is simple and straightforward. Euler buckling constraints, for example, can also be included in the formulation with an additional stress constraint:

$$-\frac{\pi^2 E}{\left(\kappa L_i / r_i\right)^2} \leq \sigma_i \qquad \forall i = 1 \ldots n_e \tag{3.2}$$

where E is the modulus of elasticity, κ is the column effective length factor, r is the member's radius of gyration ($r = \sqrt{I/A}$), and I is the member's area moment of inertia. Because Euler's buckling criterion overestimates the buckling strength of structural members, better criteria and safety factors should be considered (AISC, 2011).

3.1.2 Load–path formulation

The *load–path formulation*, also called performance index or Michell's number (Lev, 1981; Mazurek et al., 2011), has equal treatment of compression and tension members. This number takes into account the distance the internal forces travel through the structure:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{x}} \quad & Z = \textstyle\sum_i \left(N_{(t)i} - N_{(c)i}\right) L_i = \sum_i |N_i| \, L_i \\ \text{s.t.} \quad & \textstyle\sum_i A_i L_i \leq \bar{V} \\ & \mathbf{A} \geq \mathbf{0} \\ \text{with} \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \, , \end{aligned} \tag{3.3}$$

where N_i stands for the axial force in member i, and $\bar{V}$ is a prescribed limit on the volume.

The load–path formulation, has the difficulty of an absolute value in the objective function. Alternatively, the compression and tension loads can be split into two positive variables thus making the objective a linear function (Hemp, 1973). The biggest advantage of this formulation is that for statically determinate trusses, the axial load does not depend on the

cross–sectional areas. In other words, the member sizing problem is decoupled from the geometry, reducing the design variables from $(n_d * n_n + n_e)$ to just $(n_d * n_n)$, where n_d, n_n and n_e are the problem's dimension, number of nodes and number of elements respectively. The formulation can be extended to treat compression and tension differently by introducing a parameter $\gamma = -\sigma_t/\sigma_c$, and rewriting the objective for Equation (3.3) to include this penalization parameter (Sokół, 2011)

$$\begin{aligned} \min_{\mathbf{A},\mathbf{x}} \quad & Z^\star = \textstyle\sum_i \left(N_{(t)i} - \gamma N_{(c)i}\right) L_i \\ \text{s.t.} \quad & \textstyle\sum_i A_i L_i \leq \bar{V} \\ & \mathbf{A} \geq \mathbf{0} \\ \text{with} \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \end{aligned} \tag{3.4}$$

Including a buckling constraint, on the other hand, is not straightforward.

3.1.3 Compliance formulation

Recent works on structural optimization revolve around measures of structural stiffness for the structure's performance (Bendsøe and Sigmund, 2003). A typical formulation for this purpose is the *compliance formulation*: an energy measure, related to maximizing the stiffness of the structure for given loads:

$$\boxed{\begin{aligned} \min_{\mathbf{A},\mathbf{x}} \quad & C = \mathbf{u}^T\mathbf{K}\mathbf{u} = \mathbf{u}^T\mathbf{f} \\ \text{s.t.} \quad & \textstyle\sum_i A_i L_i \leq \bar{V} \\ & \mathbf{A} \geq \mathbf{0} \\ \text{with} \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \end{aligned}} \tag{3.5}$$

The volume constraint is required in this case, otherwise, the stiffest structure resembles a solid block of material. The main advantage of the compliance formulation is that the objective function is self–adjoint: when computing the sensitivity, the solution to the

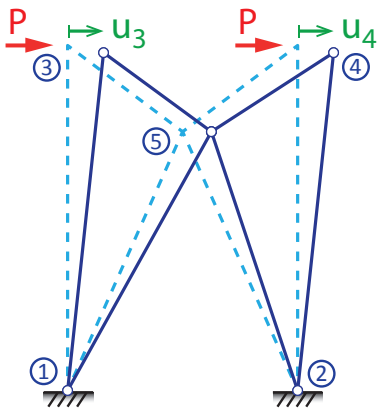


Figure 3.2: Displacements of a lateral bracing system due to a load P. The top story drift is $u_3 = u_4 = \Delta$.

adjoint problem is known (Giles and Pierce, 2000), making this formulation computationally attractive. Stress and buckling constraints can be implemented, but are, again, not straightforward.

3.1.4 Displacement formulation

Displacement objective functions are employed so as to minimize the maximum displacement, the top story displacement of a building, or inter–story drift to name a few. Considering a single displacement $\Delta = |u_j|$ (Figure 3.2), the *displacement formulation* takes the form:

$$\boxed{\begin{array}{lll} \min\limits_{\mathbf{A},\mathbf{x}} & \Delta = u_j \\ \text{s.t.} & \sum_i A_i L_i \leq \bar{V} \\ & \mathbf{A} \geq \mathbf{0} \\ \text{with} & \mathbf{K}\mathbf{u} = \mathbf{f} \end{array}} \tag{3.6}$$

The displacement formulation is simple and has the advantage of possessing a direct physical meaning for the engineer. It has similar characteristics to the compliance formulation, but it is not self–adjoint.

3.2 Formulation equivalency

The four objectives presented in Equations (3.1), (3.3), (3.5) and (3.6) may seem different, but under *typical* conditions, these formulations will result in the same optimal brace point location. In other words, the stiffest structure in a direction, the least compliant structure, the least weight structure and the one with the smallest load–path all have the same optimum solution. The focus of this section is to explain when this occurs, and what happens when additional constraints are introduced to the problem.

Optimal structures, in the sense of material efficiency, tend to be fully stressed (Michell, 1904; Lev, 1981; Topping, 1983). The proof is intuitive with the formulation in Equation (3.1): for a structure that is not fully stressed, a reduction of the cross–sectional areas will decrease the objective without violating the constraints. With no displacement, buckling or symmetry constraints (manufacturing constraints), the optimal design for a single load case is fully stressed. This statement is not true for the case of multiple loading conditions, and is not considered in the present work. The fully stressed condition leads to the "Stress–ratio method" (that relates to Michell's solutions (Lev, 1981)), where the cross–sectional areas are updated at each iteration as

$$A_{i(new)} = \max\left(\frac{\sigma_c}{\sigma_i}, \frac{\sigma_t}{\sigma_i}\right) A_i \tag{3.7}$$

Typically, optimal structures for a single load case are *statically determinate*. However, for the case of multiple loading scenarios, as Schmidt (1962) correctly concluded, *a statically indeterminate form could sometimes give a lighter structure than a statically determinate one.*

3.2.1 Load–path to volume

The connection between volume and load–path was pointed out by Michell (Lev, 1981). If the structure is fully stressed, then there are limit stress values σ_c and σ_t such that

$$N_{(c)i} = \sigma_c A_i \qquad N_{(t)i} = \sigma_t A_i \tag{3.8}$$

With these, the objective in Equation (3.4) becomes

$$Z^\star = \sum_i \left[\sigma_t \; A_i|_{(t)} - \left(-\frac{\sigma_t}{\sigma_c} \right) \sigma_c \; A_i|_{(c)} \right] L_i = \sigma_t \sum A_i L_i = \sigma_t \mathbf{A}^T \mathbf{L} \; , \tag{3.9}$$

and Equation (3.3) is a sub–case of the previous. Therefore, if the structure is fully stressed, minimizing the performance index is equivalent to the formulation in Equation (3.3) (multiplied by a constant).

3.2.2 Compliance to load–path

The compliance problem in Equation (3.5) with a single load P, independent of the design variables and displacements, can have the objective simplified as

$$C = \mathbf{u}^T \mathbf{f} = P\Delta \; , \tag{3.10}$$

where Δ is the displacement in the direction of the load P. For a truss with a single point load, the principle of work and energy (Baker, 1992) states that

$$P\Delta = \sum_i \frac{N_i^2 L_i}{A_i E} = \sum_i \frac{|N_i|^2 \, L_i}{A_i E} \tag{3.11}$$

Including the fully stressed assumption with $|N_i|/A_i = \bar{\sigma}$, results in

$$C = \sum_i \frac{|N_i|^2 L_i}{A_i E} = \frac{\bar{\sigma}}{E} \sum_i |N_i| L_i \,, \tag{3.12}$$

thus making it equivalent to the formulation in Equation (3.3) (multiplied by a constant).

3.2.3 Displacement to compliance

If the displacement problem from Equation (3.6) has a single constant load P (independent of design variables and displacements), and the displacement Δ being minimized, is in the direction of the force P, then

$$\min \Delta = \min P\Delta = \min \mathbf{u}^T \mathbf{f} \,, \tag{3.13}$$

leading to the formulation in Equation (3.5). If the objective function in Equation (3.6) is a linear combination of several displacements of the truss, then the equivalency is preserved *if and only if* the loads in the displacement directions are in the same ratio as the coefficients in the linear combination. If the previous condition is not met, then the optimal design, which minimizes some displacement Δ (or linear combination of displacements), may not be fully stressed.

3.2.4 Equivalence summary

The relationship and requirements for equivalency between formulations are then summarized in Figure 3.3. The requirements for equivalency shed light on when these connections may be broken. In particular, previous studies have already shown situations where the minimum weight and fully stressed design may not be equivalent (Kicher, 1966; Razani, 1965).

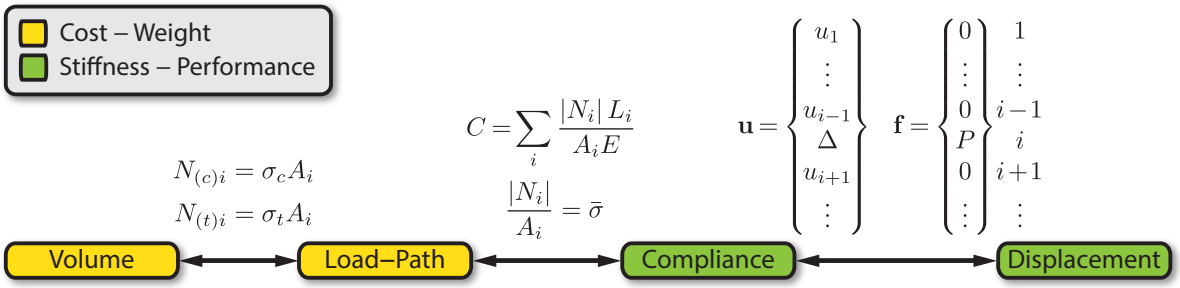


Figure 3.3: Equivalency requirements between formulations.

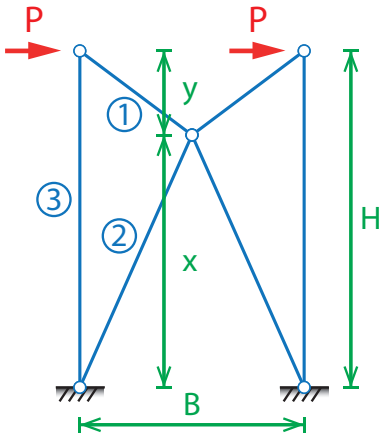


Figure 3.4: Two–dimensional lateral bracing system.

3.3 Single brace analysis

The following analytical derivations for two–dimensional braces extend the conclusions derived by Stromberg et al. (2012), where the optimal bracing point was found to be at $x = 0.75H$ for two–dimensional single–bay multiple–stories braces optimized for compliance. In addition, Stromberg et al. (2012) includes real building applications of the concepts and conclusions in the work. The findings from this single brace analysis form the basis for more general bracing rules in the following sections. The bracing point location in Figure 3.4 will be optimized using formulations discussed earlier.

Using the symmetry condition, only half of the brace needs to be analyzed. Consideration of different values for σ_t and σ_c is not important because the direction of lateral loads in buildings is uncertain, therefore a single value $\bar{\sigma}$ is used to limit positive and negative stresses.

The compliance of the structure is:

$$C = \frac{4P^2}{EB^2}\left[\frac{L_1^3}{A_1} + \frac{L_2^3}{A_2} + \frac{L_3}{A_3}y^2\right] , \tag{3.14}$$

where B denotes the bay width (Figure 3.4). The volume for the half–structure is:

$$V = \sum_{i=1}^{3} A_i L_i = A_1 L_1 + A_2 L_2 + A_3 H , \tag{3.15}$$

where H denotes the bay height (Figure 3.4). The derivatives of the member lengths L with respect to variable y are:

$$\begin{aligned} \frac{dL_1}{dy} &= \frac{y}{L_1} \\ \frac{dL_2}{dy} &= \frac{y-H}{L_2} \\ \frac{dL_3}{dy} &= 0 , \end{aligned} \tag{3.16}$$

and derivatives of the axial loads with respect to the variable y are:

$$\begin{aligned} \frac{dN_1}{dy} &= \left(\frac{-2P}{B}\right)\left(\frac{y}{L_1}\right) \\ \frac{dN_2}{dy} &= \left(\frac{2P}{B}\right)\left(\frac{y-H}{L_2}\right) \\ \frac{dN_3}{dy} &= \frac{2P}{B} , \end{aligned} \tag{3.17}$$

where N is the axial load on the member, P is the load at the top of the braced module, and x and y locate the bracing point.

The three–dimensional case is symmetric with respect to the x_1x_3 and x_2x_3 planes (lateral forces in a building are often considered in one direction at a time); therefore only one quarter of the brace needs to be solved. Figure 3.5 illustrates the bracing system loaded in the plane x_1x_3. If the base is square $B_1 = B_2$, the resulting optimal braces in both planes are the same. Corner columns participate in both loading directions, making these members attractive, if the goal is to strengthen the structure given a limited amount of material. The diagonals

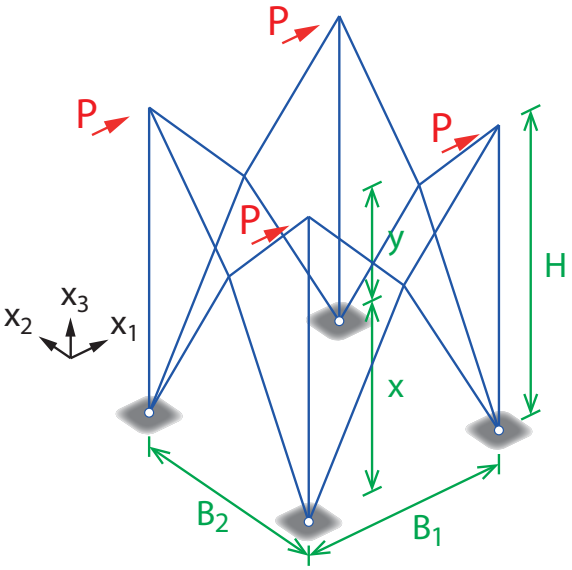


Figure 3.5: Three–dimensional lateral bracing system.

however, due to the symmetry constraint, are mirrored to the other loading plane. This can be interpreted as the *cost* of the diagonals being twice that of the two–dimensional case: a cost (or multiplicity) variable $\alpha = 1$ for the two–dimensional case and $\alpha = 2$ for the three–dimensional case will be used.

3.3.1 Minimum volume optimal

The Lagrangian for the minimum volume objective is:

$$\begin{aligned}\mathcal{L} = \alpha A_1 L_1 + \alpha A_2 L_2 + A_3 H + \lambda_{11}\left(-A_1\bar{\sigma} - N_1\right) + \lambda_{12}\left(-A_1\bar{\sigma} + N_1\right) + \ldots \\ \lambda_{21}\left(-A_2\bar{\sigma} - N_2\right) + \lambda_{22}\left(-A_2\bar{\sigma} + N_2\right) + \ldots \\ \lambda_{31}\left(-A_3\bar{\sigma} - N_3\right) + \lambda_{32}\left(-A_3\bar{\sigma} + N_3\right)\end{aligned} \tag{3.18}$$

Which has a single feasible optimum at:

$$\begin{aligned} \lambda_{11} &= \alpha L_1/\bar{\sigma} & \lambda_{12} &= 0 \\ \lambda_{21} &= 0 & \lambda_{22} &= \alpha L_2/\bar{\sigma} \\ \lambda_{31} &= 0 & \lambda_{32} &= H/\bar{\sigma} \end{aligned} \tag{3.19}$$

Therefore, the optimal bracing point is located at:

$$x = \frac{2\alpha+1}{4\alpha}H \qquad y = \frac{2\alpha-1}{4\alpha}H \tag{3.20}$$

3.3.2 Load–path optimal

The Lagrangian for the minimum load–path objective (introducing fictitious equivalent forces in the three–dimensional case to enforce symmetry, i.e. taking $\alpha = 2$) is:

$$\mathcal{L} = \alpha\frac{2PL_1}{B}L_1 + \alpha\frac{2PL_2}{B}L_2 + \frac{2Py}{B}H\,, \tag{3.21}$$

with no constraint since the structure is statically determinate. The optimal bracing point is located at:

$$x = \frac{2\alpha+1}{4\alpha}H \qquad y = \frac{2\alpha-1}{4\alpha}H \tag{3.22}$$

3.3.3 Compliance optimal

The Lagrangian for the minimum compliance objective (introducing fictitious equivalent forces in the three–dimensional case to enforce symmetry, i.e. taking $\alpha = 2$) is:

$$\mathcal{L} = \frac{4P^2}{EB^2}\left[\frac{L_1^3}{A_1} + \frac{L_2^3}{A_2} + \frac{L_3}{A_3}y^2\right] + \lambda\left(\alpha A_1 L_1 + \alpha A_2 L_2 + A_3 H - \bar{V}\right) \tag{3.23}$$

The optimum is found for

$$\lambda = \frac{4P^2y^2}{EB^2A_3^2}\,, \tag{3.24}$$

Table 3.1: Optimal bracing point location in two and three dimensions with different objectives.

Height x	Weight - Cost		Performance	
	Volume	Load–Path	Compliance	Displacement
2D	$0.75H$	$0.75H$	$0.75H$	$0.75H$
3D	$0.625H$	$0.625H$	$0.6768H$	$0.6768H$

and the optimal bracing point is at:

$$x = \frac{2\sqrt{\alpha}+1}{4\sqrt{\alpha}}H \qquad y = \frac{2\sqrt{\alpha}-1}{4\sqrt{\alpha}}H \tag{3.25}$$

3.3.4 Displacement optimal

The Lagrangian for the minimum displacement objective (introducing fictitious equivalent forces in the three–dimensional case to enforce symmetry, i.e. taking $\alpha = 2$) is:

$$\mathcal{L} = \sum \frac{N_i^2 L_i}{A_i E} + \lambda \left(\alpha A_1 L_1 + \alpha A_2 L_2 + A_3 H - \bar{V}\right) \tag{3.26}$$

The stress values in the members as a function of λ are:

$$\begin{aligned} \sigma_1 &= \sqrt{PE\lambda} \\ \sigma_2 = \sigma_3 &= \sqrt{\alpha PE\lambda}\,, \end{aligned} \tag{3.27}$$

and the optimal bracing point is found at:

$$x = \frac{2\sqrt{\alpha}+1}{4\sqrt{\alpha}}H \qquad y = \frac{2\sqrt{\alpha}-1}{4\sqrt{\alpha}}H \tag{3.28}$$

3.3.5 Results summary

Results obtained with the four objectives above are compared in Table 3.1.

The methods can be grouped (or characterized) by objectives: weight/cost and stiffness/performance. In two–dimensional braces all four formulations result in the same so-

lution. In the three–dimensional case, however, optimizing for weight/cost or for stiffness/performance results in different optimal bracing points. It is important to note that the member stresses will not be constant for the 3D performance–optimized case ($\sigma_2 = \sigma_3 = \sqrt{\alpha}\sigma_1$): the optimal design and fully stressed condition don't match (Kicher, 1966; Razani, 1965). In other words, given the symmetry constraints in two axes, the fully stressed condition is broken, and thus, the equivalency between all formulations is not maintained (Figure 3.3).

The decrease in the objective function for the optimal bracing point (as in Table 3.1), compared to the midpoint brace ($x = 0.5H$), depend on the aspect ratio of the brace. As expected, the improvement in the compliance objective (displacement decrease) with a volume constraint, is the square of the improvement in the volume objective with stress constraints, for a single two–dimensional brace:

$$\left.\frac{V(x = 0.75H, \mathbf{A}^{opt})}{V(x = 0.5H, \mathbf{A}^{opt}_{x=0.5H})}\right|_{2D} = \frac{4B^2 + 7H^2}{4B^2 + 8H^2} \tag{3.29}$$

$$\left.\frac{C(x = 0.75H, \mathbf{A}^{opt})}{C(x = 0.5H, \mathbf{A}^{opt}_{x=0.5H})}\right|_{2D} = \left[\frac{4B^2 + 7H^2}{4B^2 + 8H^2}\right]^2 \tag{3.30}$$

However, this is not true for the (symmetry–constrained) three–dimensional case. The optimization for loads in two different directions results in an optimal structure that is not fully stressed for each of the loads. The ratio of the objectives for the three–dimensional problem is as follows:

$$\left.\frac{V(x = 0.625H, \mathbf{A}^{opt})}{V(x = 0.5H, \mathbf{A}^{opt}_{x=0.5H})}\right|_{3D} = \frac{16B^2 + 23H^2}{16B^2 + 24H^2} \tag{3.31}$$

$$\left.\frac{C(x = 0.6768H, \mathbf{A}^{opt})}{C(x = 0.5H, \mathbf{A}^{opt}_{x=0.5H})}\right|_{3D} = \left[\frac{8B^2 + \left(7 + 4\sqrt{2}\right) H^2}{8B^2 + \left(8 + 4\sqrt{2}\right) H^2}\right]^2 \tag{3.32}$$

A comparison of the improvements in the objective function for a single brace, for the cases where ($B = H$) and ($1.5B = H$), is summarized in Table 3.2.

Table 3.2: Single brace improvement in the objective function for the optimal bracing compared to a mid–height bracing point.

Improvement	Weight — Cost		Performance	
over $x = 0.5H$	$B = H$	$1.5B = H$	$B = H$	$1.5B = H$
2D	8.33%	10.23%	15.97%	19.41%
3D	2.50%	3.21%	9.02%	11.28%

3.4 Multiple bays/stories

The previous section deals with a single brace loaded laterally, but in some applications the bracing system may span several stories or bays (side by side) as in Figure 3.6. Additionally, the braces could also be loaded vertically by a load P_z, but this load is only taken downwards as opposed to the lateral P_x that may act in any direction.

The optimal solution has a different bracing point for each case. Nonetheless, a unique bracing point for the whole bracing system is desirable for construction and aesthetic reasons. In this section, the optimal bracing point for several different cases are found assuming a single optimal bracing point for all modules. For statically indeterminate trusses, the cross–sectional areas must be included in the optimization.

3.4.1 Single bay — Multiple stories

Single bay braces (several stories high) as in Figures 3.6(a) and 3.6(c) are statically determinate, and the optimal bracing point using the Load–path formulation requires no member sizing. The addition of vertical loads does not have an effect on the optimal bracing point location. Vertical loads transfer directly to the base through the columns, and therefore affect only the column sizing. The optimal is found to be at $x = 0.75H$ using all of the presented formulations.

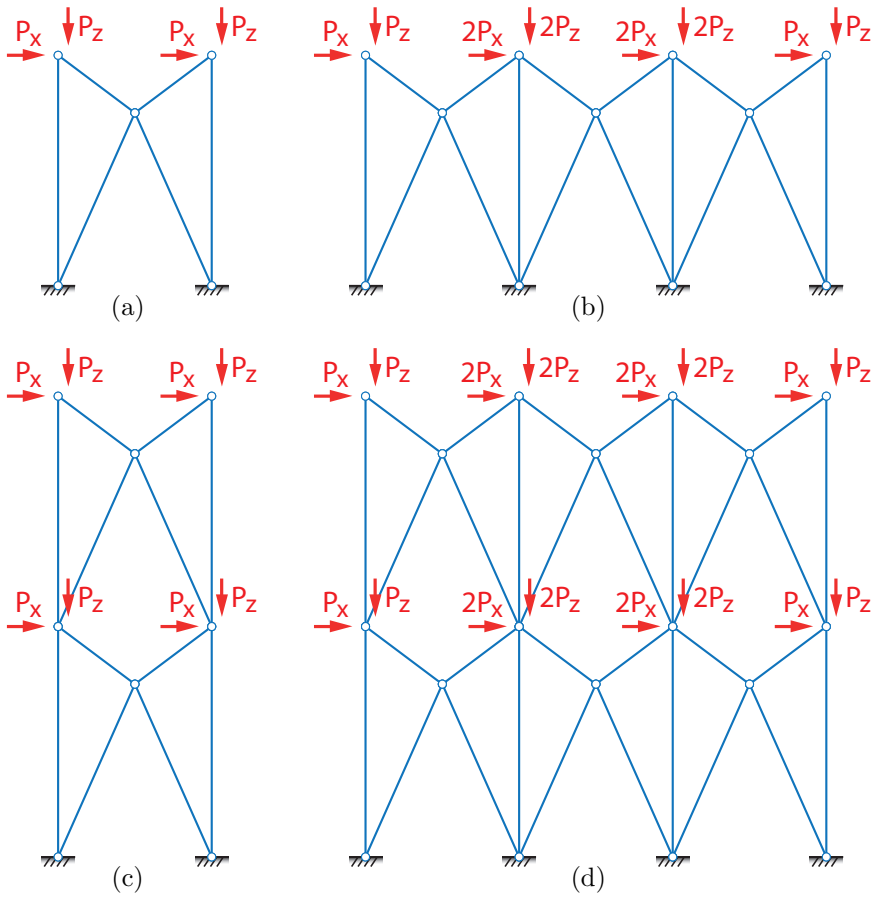


Figure 3.6: Two–dimensional bracing systems consisting of multiple bays and stories with horizontal and vertical loads. (a) 1×1 brace. (b) 3×1 brace. (c) 1×2 brace. (d) 3×2 brace.

3.4.2 Limit case of infinite bays — Single story

The bracing system is statically indeterminate in this case, and the cross–sectional areas must be included in the optimization. However, taking advantage of the symmetry, the analysis can be done as in Figure 3.7 for a single braced column. The optimum bracing points are found to be at $x = 0.50H$ using all of the presented formulations. The columns get sized with a zero cross–sectional area, and the addition of vertical loads does not change the location of the optimum bracing points. This can be interpreted as a shear transfer problem across stories, and as expected, the optimal solution consists of straight diagonal braces from the load point to the supports, as depicted in Figure 3.8. The solution for

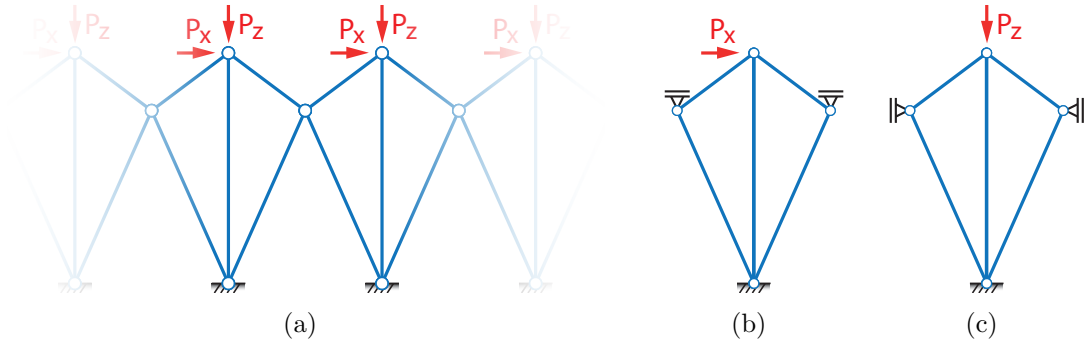


Figure 3.7: Two–dimensional single story bracing system with infinite bays. (a) Brace with loads. (b) Load and boundary conditions for horizontal load. (c) Loads and boundary conditions for vertical load.

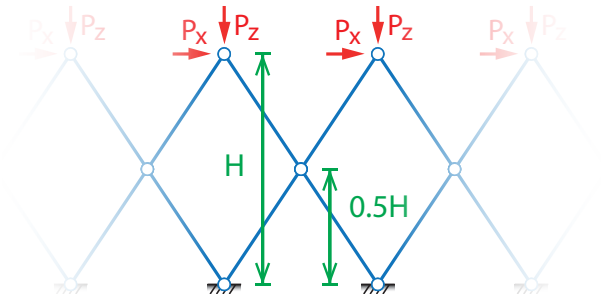


Figure 3.8: Two–dimensional optimal single story bracing system with infinite number of bays.

vertical loads only are columns sized accordingly with no diagonals (an unstable solution). Any small lateral load will cause the solution to have diagonals only and no columns.

3.4.3 Multiple bays — Multiple stories

The solution to this problem (Figure 3.6(d)) is not trivial due to the large number of variables introduced by the cross–sectional areas and the nonlinearity of the problem, and therefore the problem is solved numerically. The stress ratio method introduced in Equation (3.7) tends to drive the solution to a local minimum for large enough problems. The cross–sectional areas are introduced into the optimization along with the bracing point variable, and the optimization is done using the interior–point method (Karmarkar, 1984; Wright,

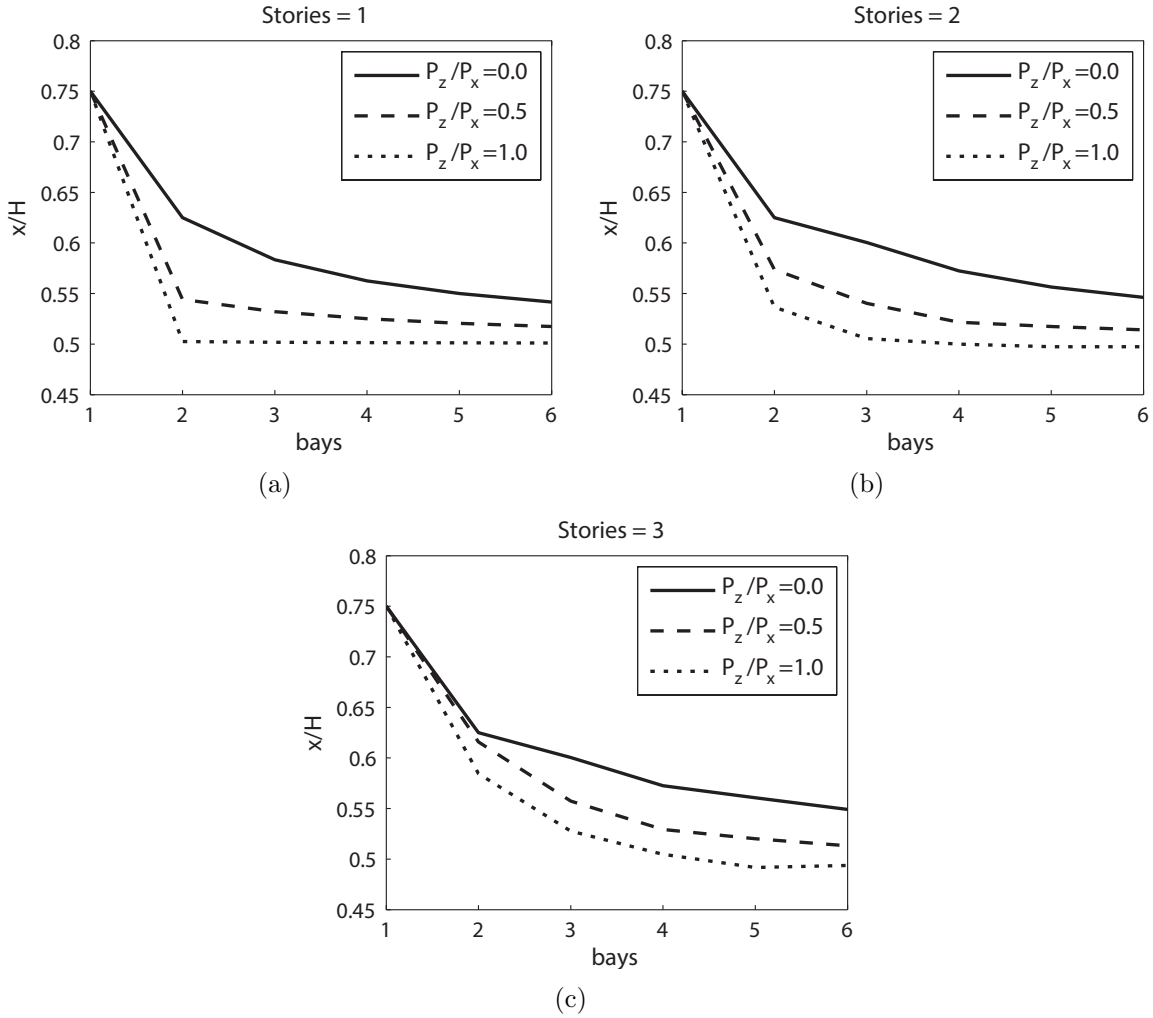


Figure 3.9: Multiple bays — Multiple stories optimal bracing locations in two dimensions. (a) One story high. (b) Two stories high. (c) Three stories high.

2004). Based on tributary areas, the lateral and vertical loads double at internal nodes. For low levels of vertical loads, the optimal solution lies between the previous solutions $x = 0.50H$ and $x = 0.75H$, and these values act like upper and lower bounds of all possible solutions. This is proven false when a bracing system is subjected to high vertical loads and multiple stories, as the solution may fall below $x = 0.5H$ and slowly converge to this value from below. From Figure 3.9, it can be inferred that an increase on the number of bays, or on the vertical loads will drive the solution closer to $x = 0.5H$. In general, the bracing point location decays asymptotically towards $x = 0.5H$. As an example, the solution for the case

with no vertical loads $P_z = 0.0$ and a single story is precisely described by

$$x_{[\text{1-story } P_z=0.0]} = \left(\frac{0.25}{bays} + 0.5 \right) H \ , \tag{3.33}$$

and on the limit of infinite bays converges to $x = 0.5H$ as predicted.

The optimal bracing point, cross–sectional areas and member stresses are given for modules with $1.5B = H$ in Figure 3.15, to serve as examples.

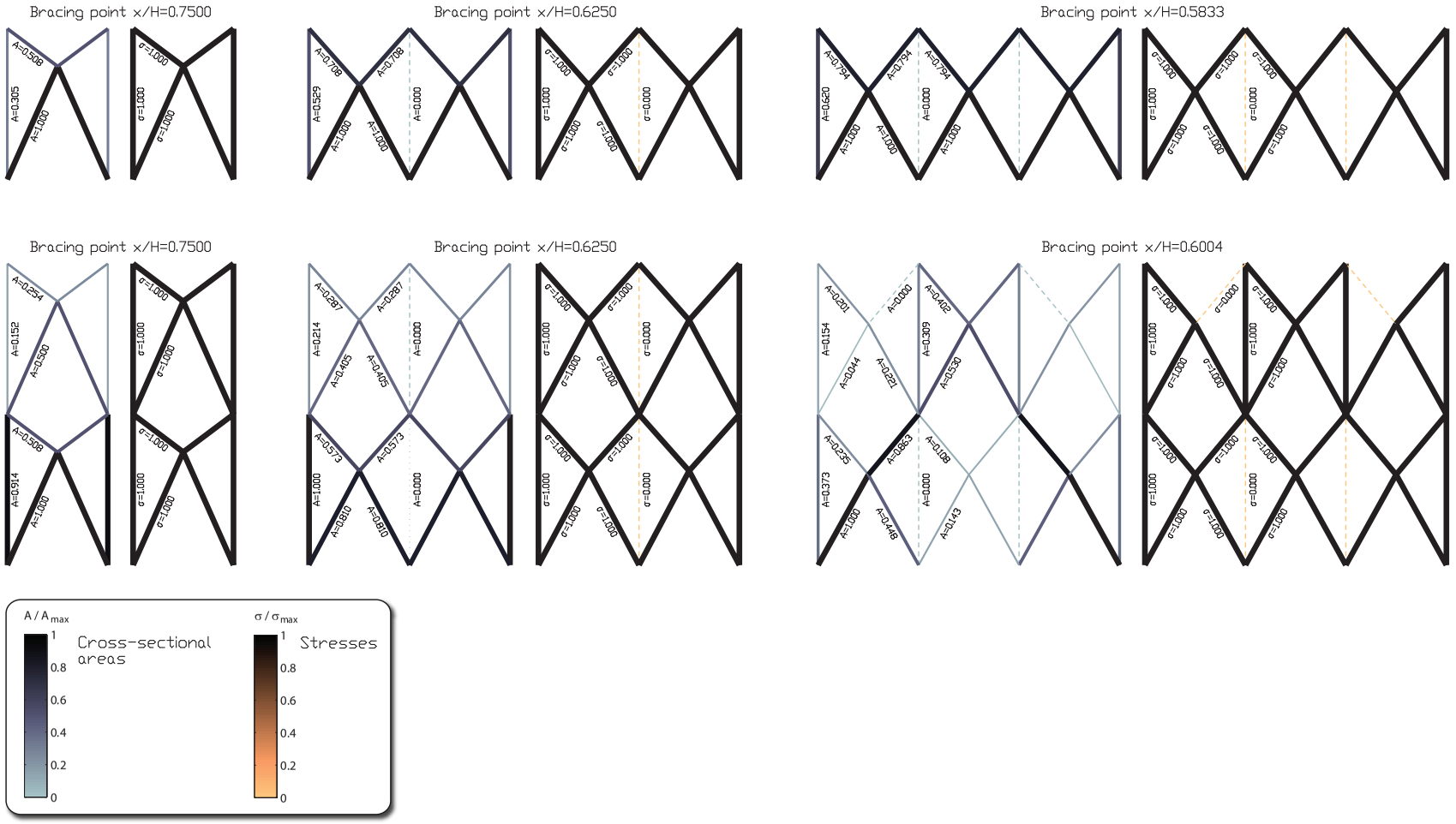


Figure 3.10: Two–dimensional optimal braces with cross–sectional areas and stresses for modules with $1.5B = H$ (areas and stresses are normalized). Symmetry is enforced and dashed members have near–zero cross–sectional area.

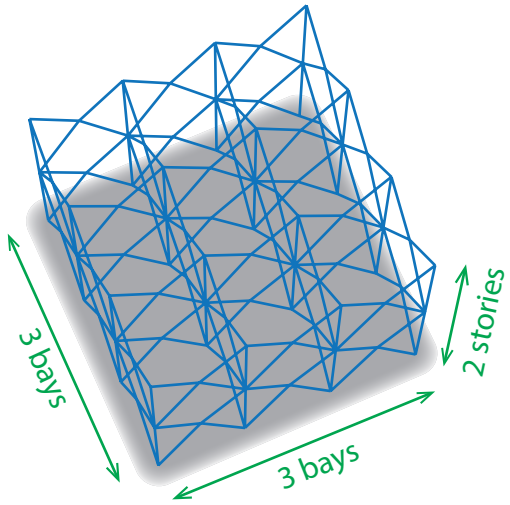


Figure 3.11: Three–dimensional brace with three bays and two stories (potential uses: stage supports, machine supports, mechanical floors, warehouses, etc).

3.4.4 Three–dimensional case

Three–dimensional braces composed of several bays have their use, for example, in mechanical floors of buildings and machine supports (Figure 3.11). Based on tributary areas and compared to corner nodes, loads double at edge nodes, and quadruple at interior nodes. The solutions are similar to the two–dimensional case, but with different upper bounds: for cost/weight optimization the upper bound is at $x = 0.625H = 5/8H$, and for stiffness/performance the upper bound is at $x = 0.6768H$ for $P_z = 0.0$, but this upper limit gets reduced with the addition of vertical loads (Figure 3.12).

The optimal bracing point, cross–sectional areas and member stresses are given for modules with $1.5B = H$ in Figures 3.16, 3.17 and 3.18, to serve as examples.

3.4.5 Extension to non–square three–dimensional braces

The analysis of three–dimensional braces is limited to square bases to narrow the scope of this section. However, the extension to braces with non–square bases follows from this work. As an example, the optimal bracing point for a single non–square three–dimensional brace,

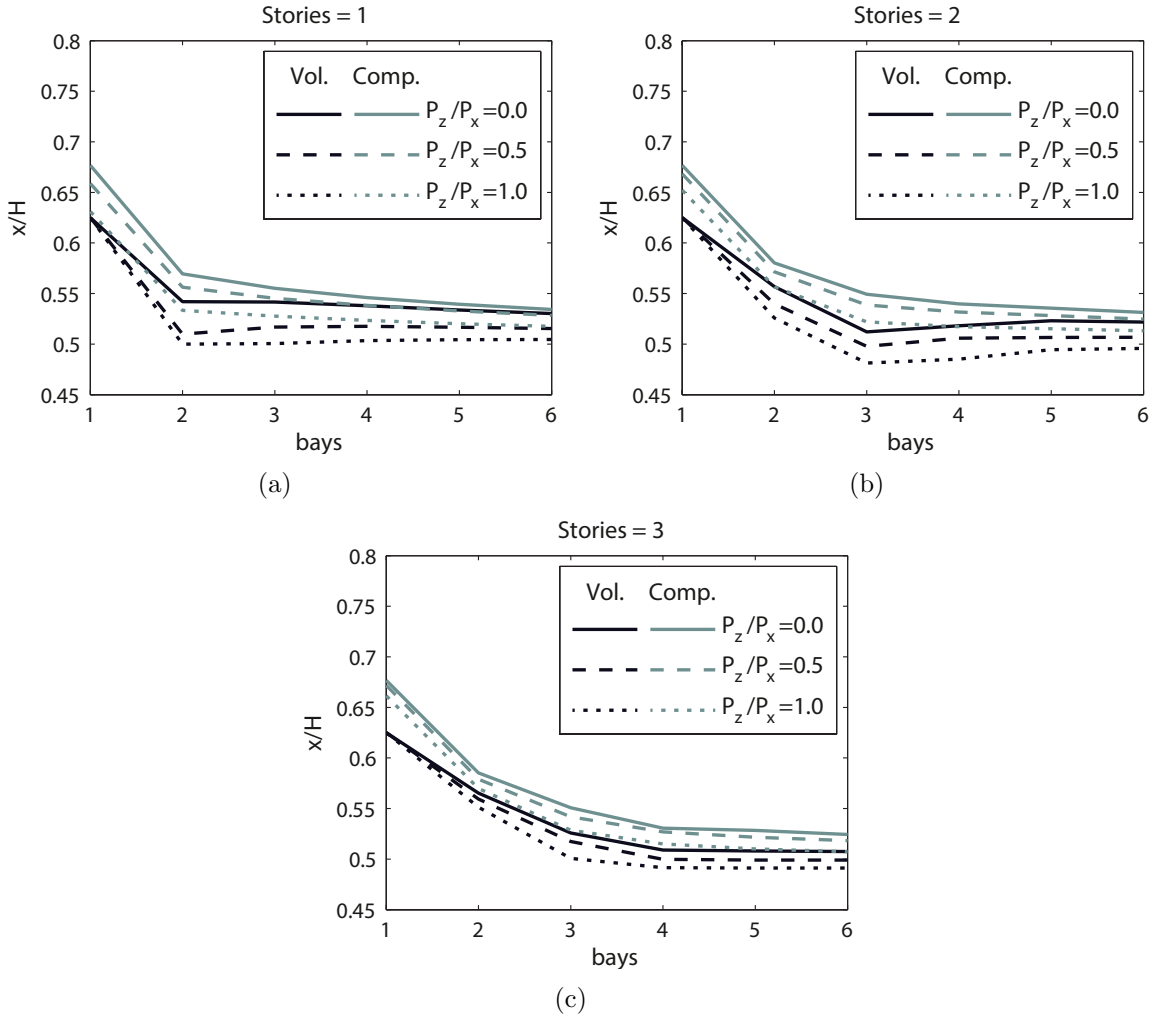


Figure 3.12: Multiple bays — Multiple stories optimal bracing locations in three dimensions. (a) One story high. (b) Two stories high. (c) Three stories high.

optimized for volume/cost, considering different loads in each direction P_1 and P_2 is:

$$x = \frac{H}{4}\left[2 + \frac{\max(\lambda_1, \lambda_2)}{\lambda_1 + \lambda_2}\right] \qquad y = \frac{H}{4}\left[2 - \frac{\max(\lambda_1, \lambda_2)}{\lambda_1 + \lambda_2}\right], \tag{3.34}$$

with $\lambda_1 = P_1/B_1$, $\lambda_2 = P_2/B_2$, and the width of the bay in the x_1 and x_2 directions being B_1 and B_2 respectively (Figure 3.5).

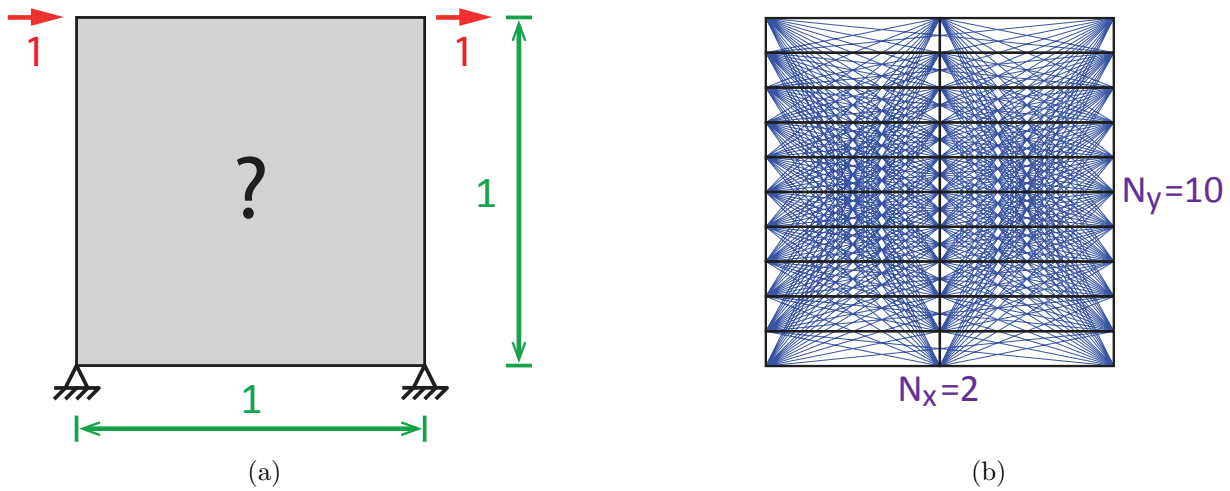
3.4.6 Additional verification with the "Ground structure method"

The numerical method used in the previous sections sizes the structural members and optimizes the geometry concurrently. The addition of the geometrical optimization makes the problem computationally and mathematically more complex than just member sizing. The ground structure method (Dorn et al., 1964; Hemp, 1973), formulates the minimum volume problem in Equation (3.1) as a Linear Program (LP), and the solution is therefore known to be globally optimal and can be computed with great computational efficiency (Karmarkar, 1984; Wright, 2004).

The power of the method relies on a highly redundant and interconnected structure (Figure 3.13). This compensates for the fact that the geometry is not really being optimized. The layout optimization problem (sizing and geometry), gets translated into a sizing–only problem, but involving a larger number of design variables. This sizing–only problem, despite its large number of design variables, it is computationally more efficient and globally optimal. Using an implementation of the method developed for unstructured domains (explained in detail in Chapters 4 and 5), the results previously obtained for a single brace, are again confirmed numerically (Figure 3.14).

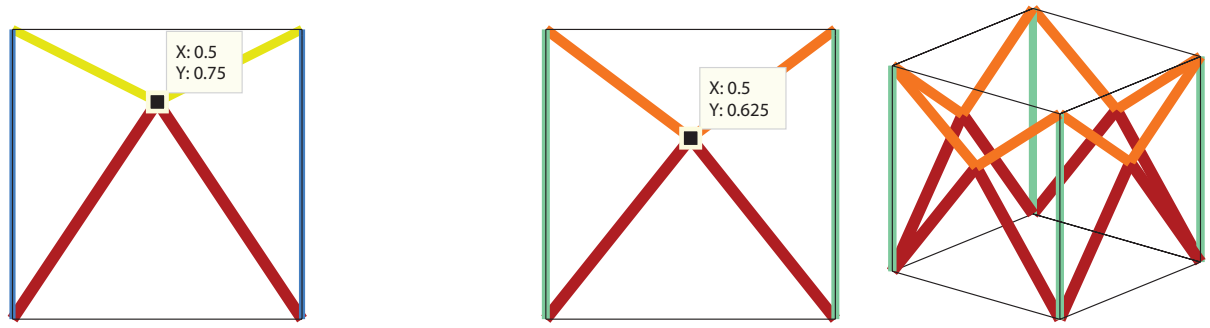
3.5 Conclusions

Optimal structures, and consequently bracing systems are said to be optimal in accordance to the objective function used. Four common formulations (using different objective functions) for structural optimization were presented. Similarities, differences and connections between them were highlighted and explored. Compared to geometrical optimization, the member sizing problem is better known and understood. Therefore, the focus of this work was placed on the bracing point location, leaving the member sizing to be done a posteriori (although, to calculate the optimal bracing point for indeterminate trusses, the members had to be



(a) (b)

Figure 3.13: Ground structure optimization of a braced module. (a) Problem definition. (b) Ground structure (interconnected truss) for a 2×10 partition.



(a) (b)

Figure 3.14: Optimized brace for minimum volume using the ground structure method using a 2×200 partition. (a) Two–dimensional optimal brace with $x = 0.75H$. (b) Three–dimensional optimal brace with $x = 0.625H$.

sized).

The optimal bracing point location in two–dimensions is the same regardless of the objective function (formulation) used. In most cases, the optimal bracing point is found to be in a feasible region delimited by $x = 0.50H$ and an upper limit or bound. The optimal bracing point location may fall below $x = 0.50H$ if the structure is subjected to high vertical loads. If the bracing system is subjected to vertical loads, and/or if the bracing system is composed of multiple bays, then the optimal bracing point location approaches the lower limit. The

upper limit is $x = 0.75H = 3/4H$ for the 2D case, regardless of the objective function. For the three–dimensional case, the upper limit is $x = 0.625H = 5/8H$ for cost/weight optimized structures, and for the stiffness/performance case the upper limit is $x = 0.6768H$, or a lower value if subjected to vertical loads.

The optimal cross–sectional areas depend on the aspect ratio of the module. The symmetry constraint in three–dimensional trusses breaks the fully stressed condition in structures optimized for stiffness/performance. In other words, material is not being used efficiently (or at full capacity) but the resulting structure will still be the stiffest.

The purpose of this chapter is to provide insight for the initial guess of cost–effective or high–performing lateral braces. These findings can aid the engineer in the initial stages of design, and also provide guidance to improve common engineering practices that often put the bracing point at the middle $x = 0.5H$, or worse, at the top $x = H$. A natural extension of his work is the consideration of the dynamic behavior of the structure.

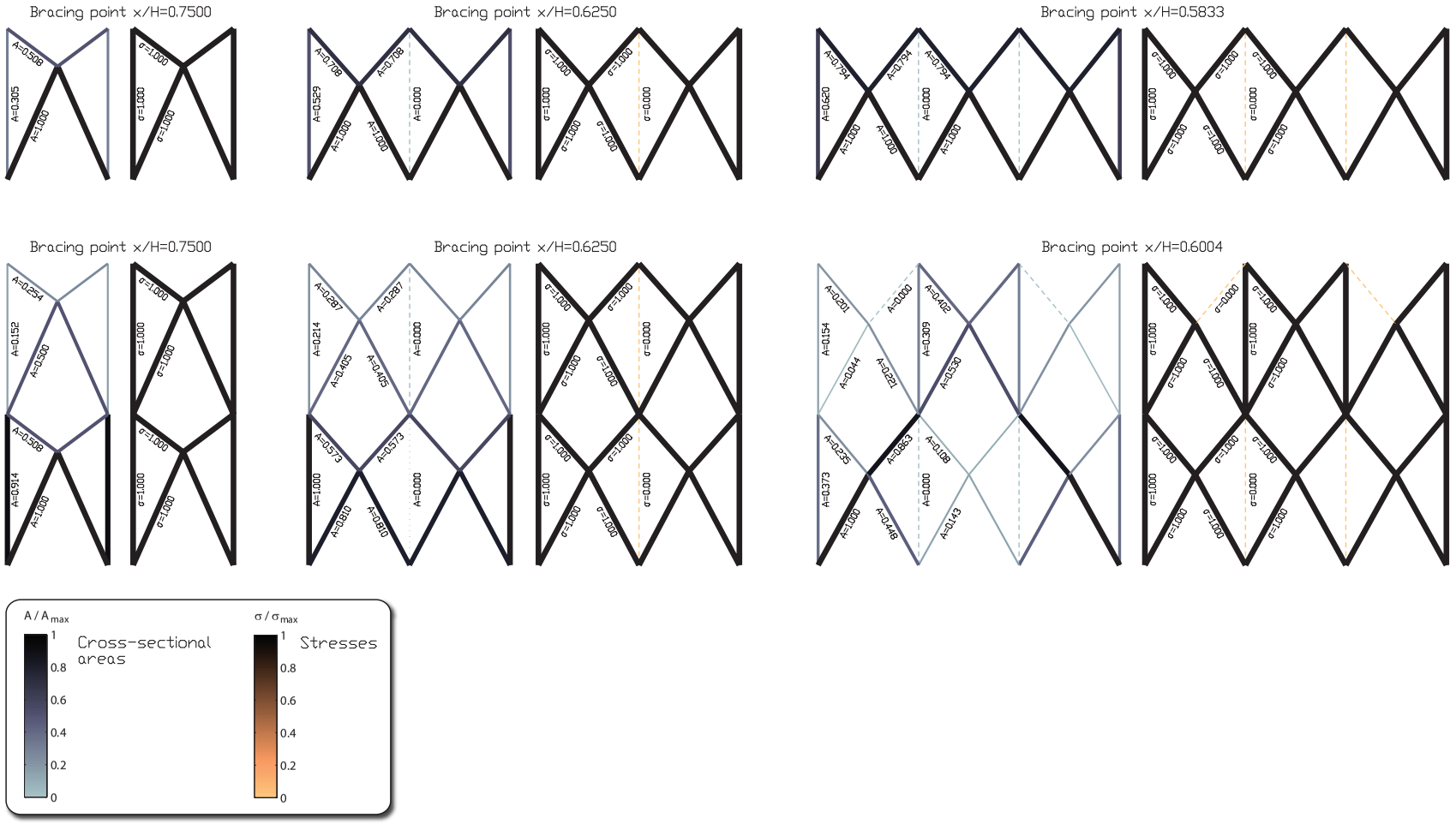


Figure 3.15: Two–dimensional optimal braces with cross–sectional areas and stresses for modules with $1.5B = H$ (areas and stresses are normalized). Symmetry is enforced and dashed members have near–zero cross–sectional area.

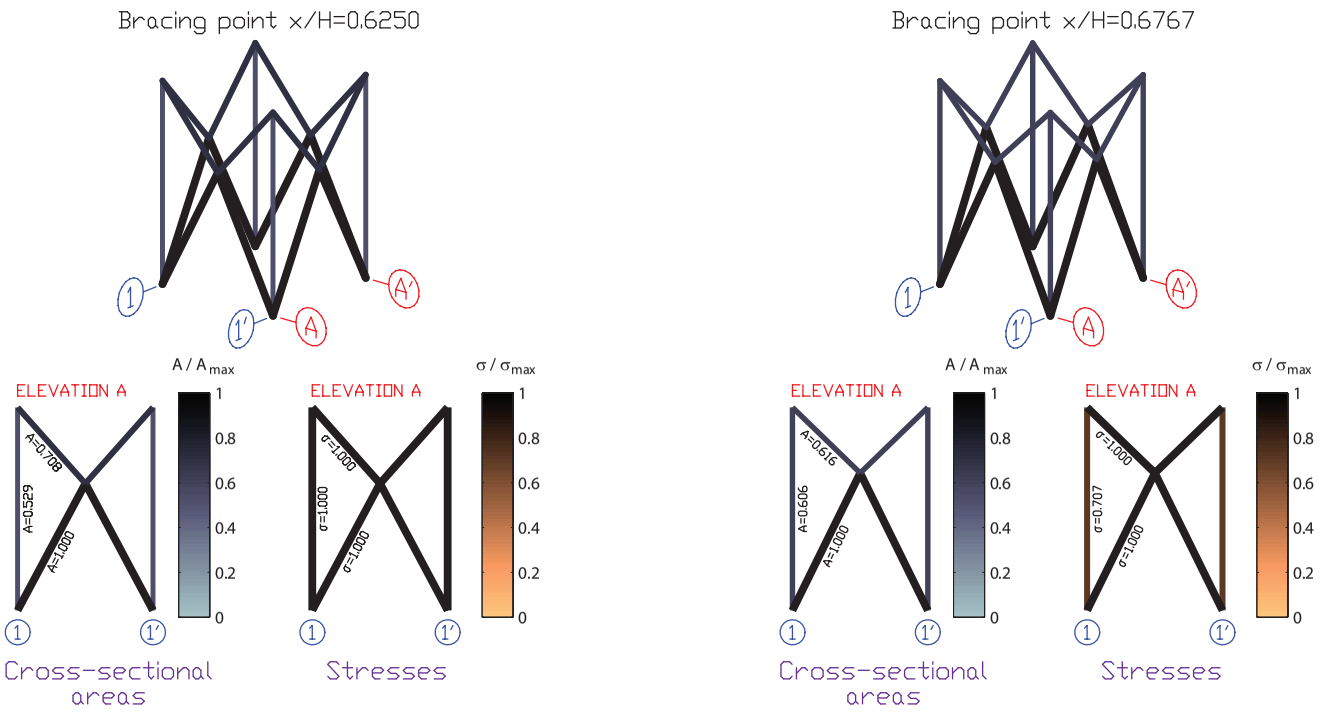
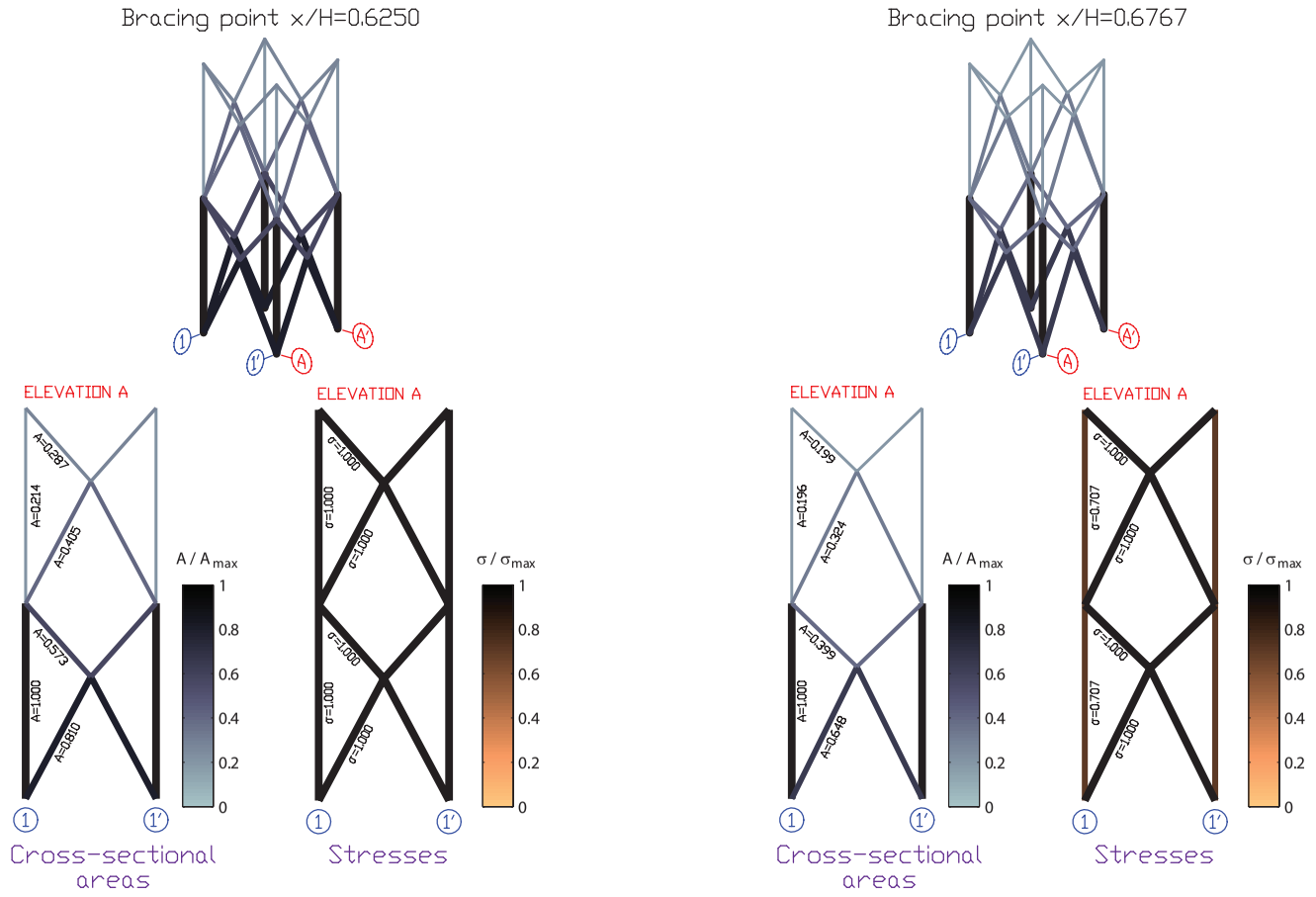


Figure 3.16: Three–dimensional optimal brace with cross–sectional areas and stresses for a 1–bay 1–story truss with $1.5B = H$ (areas and stresses are normalized). Symmetry is enforced and dashed members have near–zero cross–sectional area. (a) Optimized for volume. (b) Optimized for compliance.



(a) (b)

Figure 3.17: Three–dimensional optimal brace with cross–sectional areas and stresses for a 1–bay 2–stories truss with $1.5B = H$ (areas and stresses are normalized). Symmetry is enforced and dashed members have near–zero cross–sectional area. (a) Optimized for volume. (b) Optimized for compliance.

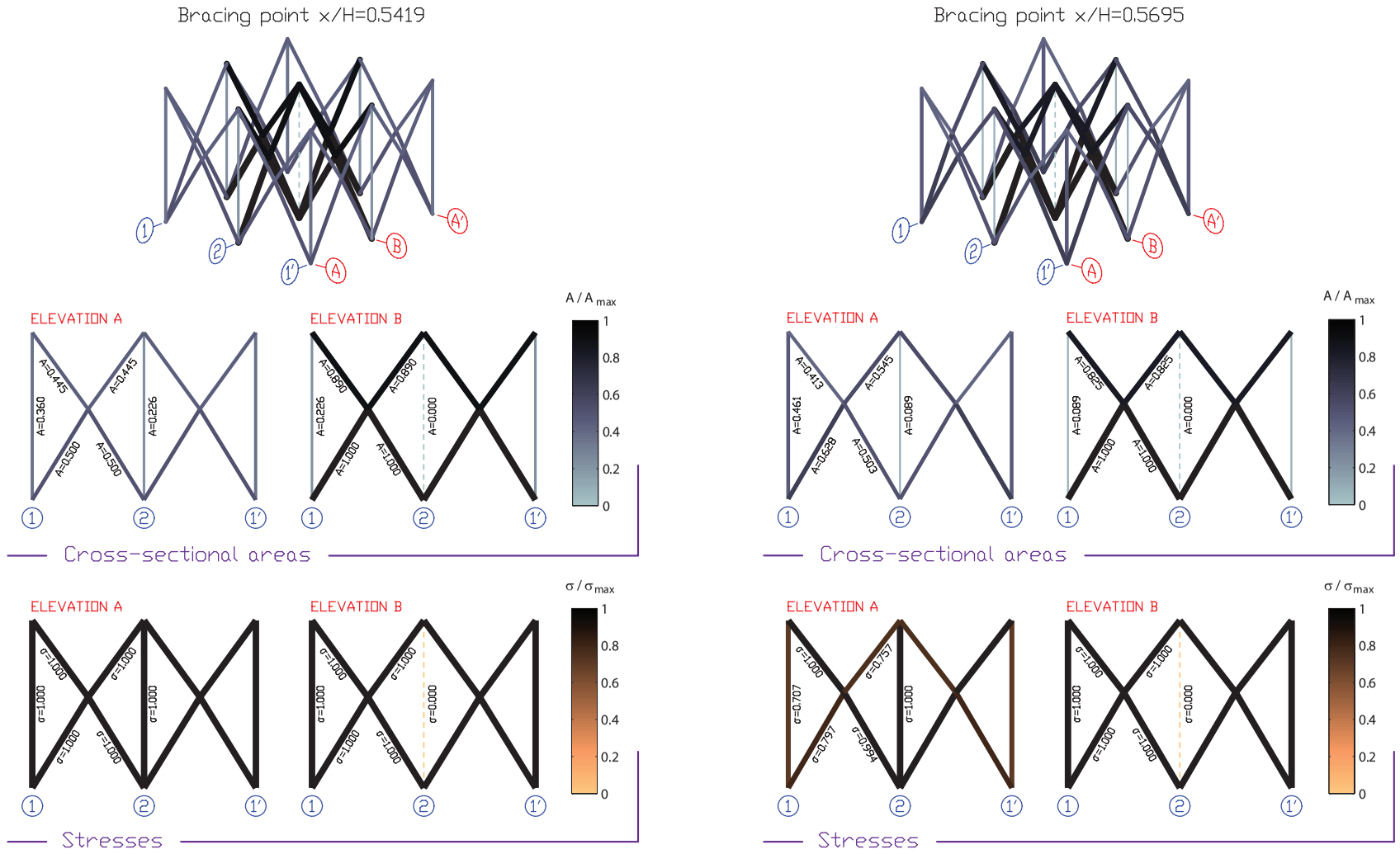


Figure 3.18: Three–dimensional optimal brace with cross–sectional areas and stresses for a 2–bays 1–story truss with $1.5B = H$ (areas and stresses are normalized). Symmetry is enforced and dashed members have near–zero cross–sectional area. (a) Optimized for volume. (b) Optimized for compliance.

Chapter 4

Unstructured ground structure method in 2D

The ground structure method (Dorn et al., 1964) provides an approximation to an optimal Michell structure (Michell, 1904; Hemp, 1973) composed of an infinite number of members, by using a reduced finite number of truss members. The optimal (least–weight) truss for a single load case, under elastic and linear conditions, subjected to stress constraints can be formulated as a linear programming problem (Ohsaki, 2010). The method removes unnecessary members from a highly interconnected truss (*ground structure*) while keeping the nodal locations fixed. Hegemier and Prager (1969) showed that a truss with maximum stiffness is also fully stressed. In addition, the problem of a single load case considering equal stress limits in compression and tension, is equivalent to the minimization of compliance for a prescribed volume (Bendsøe and Sigmund, 2003).

The analytical solution must satisfy some known conditions for structural optimization problems (Michell, 1904; Hencky, 1923). However, these conditions themselves do not provide means for obtaining the optimal analytical solution. Given a *candidate optimal structure*, these requirements can be used to check if the structure is indeed optimal or not. This is where the ground structure method excels; it provides a solution that is close to the absolute optimal solution sought.

The ground structure method has been refined, simplified and optimized, resulting in an easy–to–use implementation for truss topology optimization in structured orthogonal do-

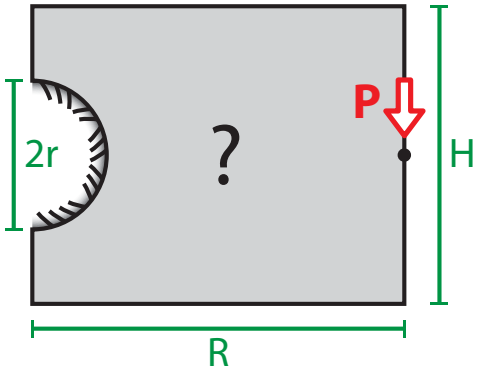


Figure 4.1: Cantilever with circular support. The analytical solution is given by Michell (1904) provided that the height H is large enough to develop the complete solution.

mains (Sokół, 2011). The method has also been extended to support unstructured meshes (Smith, 1998), where the initialization of the method (generation of the ground structure) is intricate. Recently, exact solutions for complicated domains have been numerically approximated and obtained (Lewiński et al., 2013), and there is ongoing work to extend the library of known analytical solutions for complicated domains.

The interest in unstructured non–orthogonal domains is reasonable; applied engineering problems are often not composed of *boxes*. The analytical solution of a cantilever supported on a circle (Figure 4.1) was obtained by Michell (Michell, 1904), and later generalized to different geometries and conditions (Graczykowski and Lewiński, 2005). This problem, for example, cannot be solved with an orthogonal structured domain. The present work extends the ground structure method with a simple, flexible and effective methodology to generate the ground structures in non–orthogonal unstructured and concave domains. However, the method is restricted to piecewise polygonal boundaries (convex, concave and with the possibility of holes).

The computational implementation, named GRAND, aims to have a balance between performance and legible code (educational). The objective is to provide future researchers in structural optimization with a ground structure implementation that serves as a starting point for future developments. The complete source code for GRAND is available in

Appendix A. Limitations and assumptions of the present implementation and method are:

- single static load case scenario
- constant forces (design independent)
- small deformations
- two–dimensional (2D) problems

It can, however, address different limits in tension σ_T and compression σ_C (Sokół, 2011).

Throughout this chapter, the terms *truss member* and *bar* are used interchangeably.

4.1 Formulations

Michell (1904) derived the conditions necessary for a minimum volume truss subjected to stress constraints (Ohsaki, 2010; Hemp, 1973): given stress limits in tension $\sigma_T > 0$ and compression $\sigma_C > 0$, and the average limit stress $\sigma_0 = (\sigma_T + \sigma_C)/2$, the truss is optimal if:

1. The truss is in equilibrium
2. The stress is equal to either σ_T or σ_C for all members
3. There exists a compatible deformation field such that the strains are equal to $\varepsilon_t = \sigma_0\varepsilon_0/\sigma_T$ and $\varepsilon_c = \sigma_0\varepsilon_0/\sigma_C$ for members in tension and compression, respectively

As a consequence, the members in the resulting structure are arranged in the directions of the principal strains for the displacement field. Unfortunately, Michell's solutions encompass infinitely dense members. Nonetheless, a reasonable approximation to this solution can be obtained using a (finite) large number of members within the prescribed domain.

The problem formulation used in this work is based on *plastic analysis*: no stiffness matrices, compatibility equations or stress–strain relations are used (Hemp, 1973). However, for the sake of completeness, the *elastic analysis* and some issues present in this method will also be discussed (Christensen and Klarbring, 2009).

4.1.1 Elastic formulation

Consider a rigid truss (no mechanisms), with N_{dof} nodal forces $\mathbf{f}$ (excluding the components with supports), and assume that the supports are sufficient to prevent the structure from having rigid body motions. The basic formulation for the minimum volume truss is then (Ohsaki, 2010):

$$\begin{aligned} \min_{\mathbf{a}} \quad & V = \mathbf{l}^T \mathbf{a} \\ \text{s.t.} \quad & \mathbf{K}\mathbf{u} = \mathbf{f} \\ & -\sigma_C \leq \sigma_i \leq \sigma_T \ \text{ if } \ a_i > 0 \\ & a_i \geq 0 \quad i = 1, 2 \ldots N_b \,, \end{aligned} \tag{4.1}$$

with a_i, l_i and σ_i the cross–sectional area, length and stress of the ith member (for all N_b members). The parameters N_n and N_{sup} are the number of nodes and components with supports respectively, and $N_{dof} = 2N_n - N_{sup}$ for a two–dimensional ground structure. Here, $\mathbf{K}$ denotes the global stiffness matrix and $\mathbf{u}$ denotes the nodal displacements associated with the N_{dof} free nodal components. Theoretically, a member is absent (removed) from the truss if $a_i = 0$. This issue has received significant attention in the literature, and is further discussed in the next paragraph. The redundancy of the ground structure is $N_b - N_{dof}$ and should be greater than zero to provide optional layouts.

This formulation considers the equilibrium and compatibility conditions, and is thus an *elastic analysis* formulation (Hemp, 1973; Kirsch, 1993). The stress constraint may be violated if its corresponding member is absent, i.e. $a_i = 0$ (Sved and Ginos, 1968). This phenomena of is known as *vanishing constraints* or *design–dependent constraints*. For the case of multiple loads, the optimal solution may become a *singular topology*, and thus obtaining the global optimum becomes quite challenging (Rozvany, 2001). Fortunately, the single load case does not suffer from this problem.

4.1.2 Plastic formulation

Compared to Equation (4.1), a formulation based on *plastic analysis* enforces equilibrium and no explicit compatibility or stress–strain relations (Kirsch, 1993):

$$\begin{aligned} \min_{\mathbf{a}} \quad & V = \mathbf{l}^T\mathbf{a} \\ \text{s.t.} \quad & \mathbf{B^T n} = \mathbf{f} \\ & -\sigma_C a_i \leq n_i \leq \sigma_T a_i \quad i = 1, 2 \ldots N_b \,, \end{aligned} \tag{4.2}$$

where $\mathbf{B^T}$ is the nodal equilibrium matrix of size $N_{dof} \times N_b$ ($\mathbf{B}$ has size $N_{dof} \times N_b$), built from the directional cosines of the members, and $\mathbf{n}$ is a vector with the internal (axial) force for all members in the ground structure. *The stress constraint (in tension or compression) must be active for all members at the optimum.* An intuitive proof is that if the ith member has $n_i < \sigma_T a_i$ and $n_i > -\sigma_C a_i$, then a_i can be reduced (reducing the total volume) without violating the constraints. The stress constraint is expressed in terms of member force, thus simplifying its treatment. Incorporating *slack variables* in the stress constraints (Hemp, 1973; Achtziger, 2007), converts the stress inequalities into equalities:

$$\begin{aligned} n_i + 2\frac{\sigma_0}{\sigma_C} s_i^- &= \sigma_T a_i \\ -n_i + 2\frac{\sigma_0}{\sigma_T} s_i^+ &= \sigma_C a_i \,, \end{aligned} \tag{4.3}$$

where the (positive) coefficients multiplying the slack variables simplify the resulting expressions for cross–sectional area and axial force:

$$\begin{aligned} a_i &= \frac{s_i^+}{\sigma_T} + \frac{s_i^-}{\sigma_C} \\ n_i &= s_i^+ - s_i^- \end{aligned} \tag{4.4}$$

The optimization problem in (4.2) becomes then a linear programming problem as:

$$\begin{aligned} \min_{\mathbf{s}^+,\mathbf{s}^-} \quad & V = \mathbf{l}^T \left(\frac{\mathbf{s}^+}{\sigma_T} + \frac{\mathbf{s}^-}{\sigma_C} \right) \\ \text{s.t.} \quad & \mathbf{B^T} \left(\mathbf{s}^+ - \mathbf{s}^- \right) = \mathbf{f} \\ & s_i^+, s_i^- \geq 0 \end{aligned} \tag{4.5}$$

Note that for any active member, only one of s_i^+ and s_i^- is non–zero. The member is in tension if $s_i^+ > 0$, and in compression if $s_i^- > 0$. If the truss structure is stable, has no repeated and no overlapping members, then the rank of matrix $\mathbf{B^T}$ is N_{dof} (i.e. the solution does not lie on the edge of the feasible domain). The solution of the linear programming problem (4.5) yields at most N_{dof} non–zero *basic variables*, with the remainder *non–basic variables* being absent from the optimal structure (i.e. $a_i = 0$). Therefore, the optimal truss is statically determinate and the solution is also globally optimal (Sved, 1954; Kicher, 1966). The elastic design (Equation 4.1) has to satisfy additional compatibility conditions, and thus a higher optimal volume is expected compared to the plastic design (Equation 4.2). But because the optimal structure for a single load case is statically determinate, then the *plastically admissible structure*, based on force equilibrium, also satisfies the kinematic compatibility and stress–strain relation, and is thus also an *elastically admissible structure*: the optimal solution for both methods are equal for the given assumptions (Dorn et al., 1964; Hemp, 1973).

If the ratio in the stress limits is defined as $\kappa = \sigma_T / \sigma_C$, then the formulation becomes:

$$\begin{aligned} \min_{\mathbf{s}^+,\mathbf{s}^-} \quad & \bar{V} = \frac{V}{\sigma_T} = \mathbf{l}^T \left(\mathbf{s}^+ + \kappa \mathbf{s}^- \right) \\ \text{s.t.} \quad & \mathbf{B^T} \left(\mathbf{s}^+ - \mathbf{s}^- \right) = \mathbf{f} \\ & s_i^+, s_i^- \geq 0 \end{aligned} \tag{4.6}$$

This final form of the *plastic layout optimization* problem is utilized in this work (Sokół, 2011; Achtziger, 2007; Gilbert and Tyas, 2003), and can be efficiently solved using the interior–

point algorithm (Karmarkar, 1984; Wright, 2004). The optimal volume $\bar{V}$ is calculated for $\sigma_T = 1$, and should be scaled by $1/\sigma_T$ for values other than unity.

The resulting optimal structure will be in equilibrium. However, the equilibrium may be unstable due to the existence of members with zero cross–sectional areas belonging to the *basic variables* (degenerate LP problem). As recommended by Dorn et al. (1964), the structure should be post–processed to become a *reduced optimal structure* (ROS) with:

- Nodes connecting two collinear members are all replaced with a single long member (collinear hinges).
- Nodes where all members have $a_i = 0$ are removed, i.e. nodes not participating in the resulting structural configuration are removed.
- Members associated with basic variables (LP) having zero cross–sectional area should have a minimum value $a_{min} > 0$ to account for imperfections and small variations in the geometry and loads.

The resulting (stable) structure maintains all the properties of the original one: equilibrium, statically determinate and fully stressed. This post–processing however, is left to the judgement of the user or practitioner and will not be addressed in this chapter.

4.2 Implementation

4.2.1 Domain definition — Base mesh

To define the domain and boundaries, four variables must be specified. These in turn will be used to generate the ground structure, and are described in detail in Table 4.1. The number of nodes, elements, nodes with prescribed boundary conditions and nodes with prescribed loads are N_n, N_e, N_f and N_l respectively. It should be noted that GRAND makes no assumption on the type of elements (lines, triangles, quads, polygonal or combinations of

Table 4.1: Domain definition (base mesh) input variables for GRAND.

Variable Name	Type & Size	Description
`NODE`	`array` $N_n \times 2$	Each row p has the nodal coordinates x and y for node p.
`ELEM`	`cell` $N_e \times 1$	Every element in the list is a row vector containing the node numbers for a particular element.
`SUPP`	`array` $N_f \times 3$	Each row consists of a node number, fixity x and fixity in y. Any value other than `NaN` specifies fixity. The total number of specified fixities is N_{sup}.
`LOAD`	`array` $N_l \times 3$	Each row consists of a node number, load in x and load in y. A zero or `NaN` specify no force in that direction.

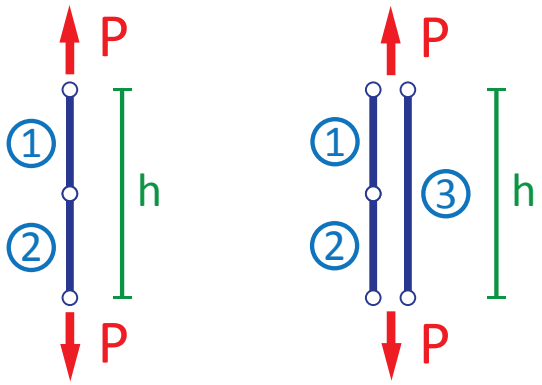


Figure 4.2: Overlapping members example assuming $P = 1$, $h = 1$ and $\sigma_T = 1$. (a) Problem with a unique solution: optimal volume is $V = 1$ and $a_1 = a_2 = 1$. (b) Problem with a non–unique solution: optimal volume is $V = 1$, but $a_1 = a_2 = [0, 1]$ and $a_3 = 1 - a_1$.

these), nor on the element numbering. This information is only used to define the nodal connectivity, the domain's extension and boundaries. GRAND has three options available for importing or generating the base mesh: loading an external mesh following the guidelines from Table 4.1, generating the mesh with the bundled polygonal mesher called PolyMesher (Talischi et al., 2012a), and generating an orthogonal structured domain using a subroutine provided with GRAND.

4.2.2 Ground structure generation

The generation of the ground structure is the main contribution in GRAND. To ensure that the solution is unique and the representation of the solution is appropriate, the ground structure should have no overlapping truss members (members connecting collinear nodes), or two members connecting the same nodes (Figure 4.2). The starting point of the ground structure generation process is the base mesh (defined as in Table 4.1). In addition, a connectivity level Lvl and a collinearity tolerance $ColTol$ need to be specified. To efficiently generate the ground structure on modern computer architectures, the problem of generating the ground structure should be translated to linear algebra and matrix operations when possible, thus taking advantage of the optimizations in numerical computing frameworks (Heath, 1998; Olson, 2013).

The user defined *connectivity level* determines the level of redundancy, or inter–connectedness, of the initial ground structure. If the connectivity level Lvl is sufficiently high, the ground structure generation algorithm will interconnect all nodes; this is often referred to as a *full level ground structure*. The analytical solution (Michell, 1904), is typically composed of curved members. The ground structure method will have a tendency to retain short members, so as to more accurately try to represent these curved members in a piecewise fashion. Thus, the full level ground structure, containing long (edge–to–edge) candidate members, is not the best from a cost–effective point of view. If two nodes belong to the same element in the base mesh (Figure 4.3(a)), then they are considered *neighbors*. From this idea of neighbors, the connectivity level can be explained as follows:

- Level 1 connectivity will generate members between all neighboring nodes as in Figure 4.3(b).
- Level 2 connectivity will generate members up to the neighbors of the neighbors as in Figure 4.3(c).
- Level 3 connectivity will generate members up to the neighbors of the neighbors of the

neighbors as in Figure 4.3(d).

- Level 4 connectivity ...

The example in Figure 4.3 achieves a *full level ground structure* at level 5, and the difference between levels 4 and 5 is minimal. Note that no members are generated in the domain's concave region; this desirable feature will be discussed in detail in the following sections. The member generation process scales rapidly with the connectivity level, only decelerates when the ground structure reaches full level connectivity (Figure 4.4).

The nodal connectivity matrix (symmetric i.e. bi–directional) for the base mesh is named

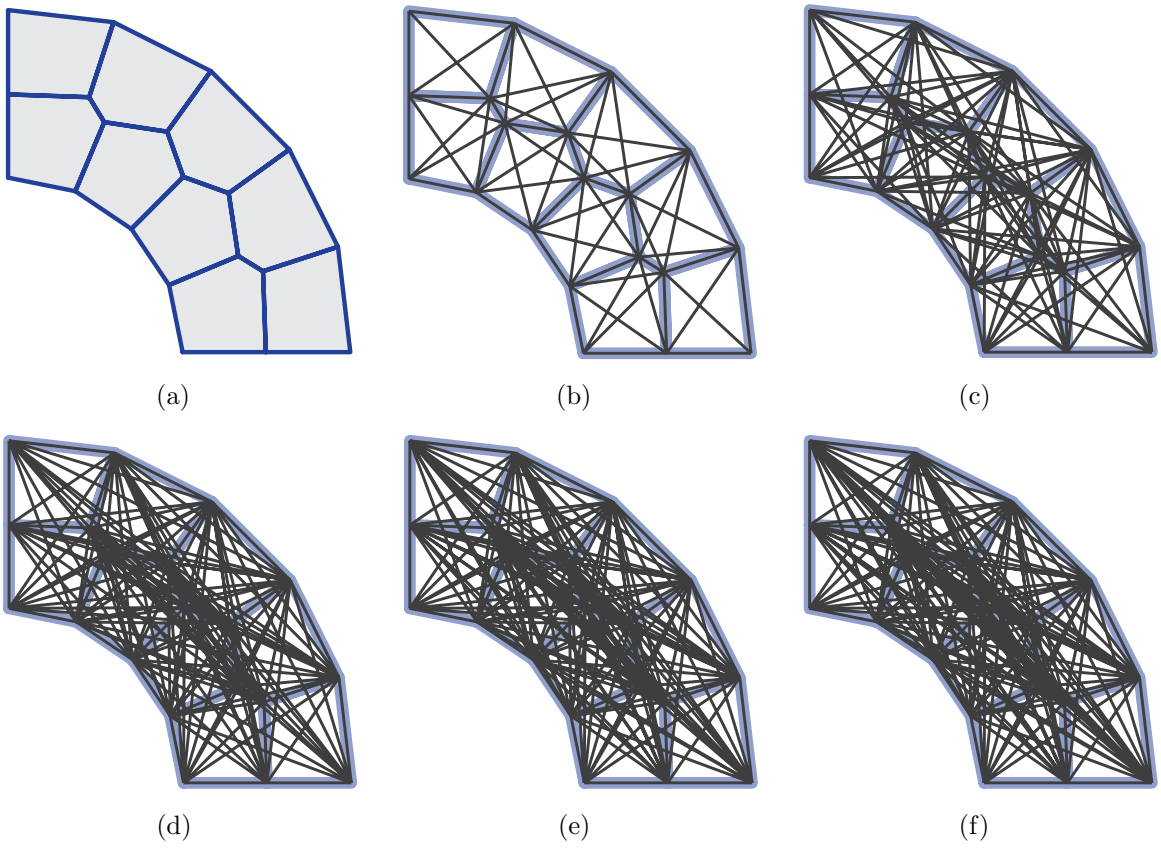
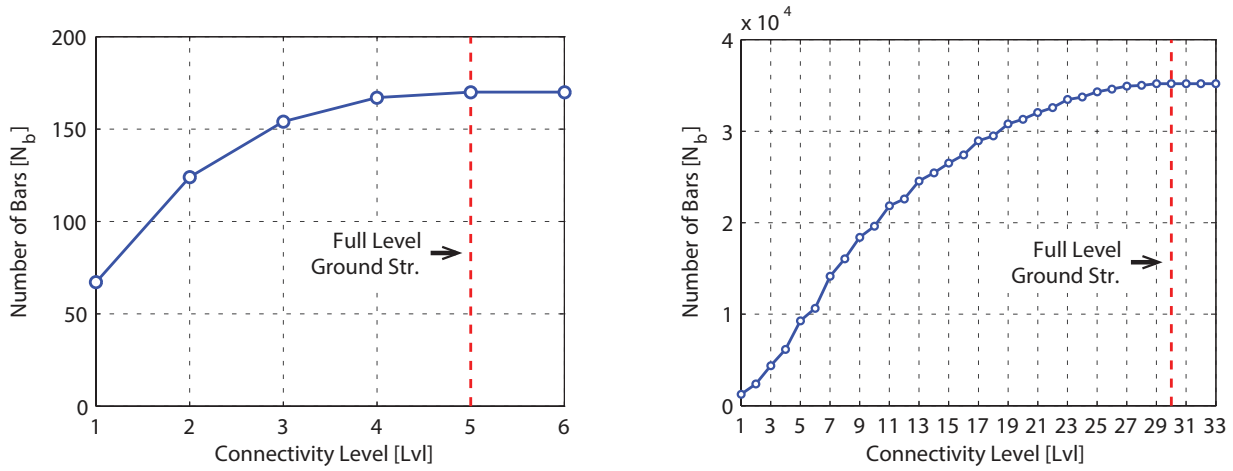


Figure 4.3: Ground structure connectivity level generation example. (a) Base mesh composed of 9 polygonal elements. (b) Level 1 connectivity. (c) Level 2 connectivity. (d) Level 3 connectivity. (e) Level 4 connectivity. (f) Level 5 connectivity.



(a) (b)

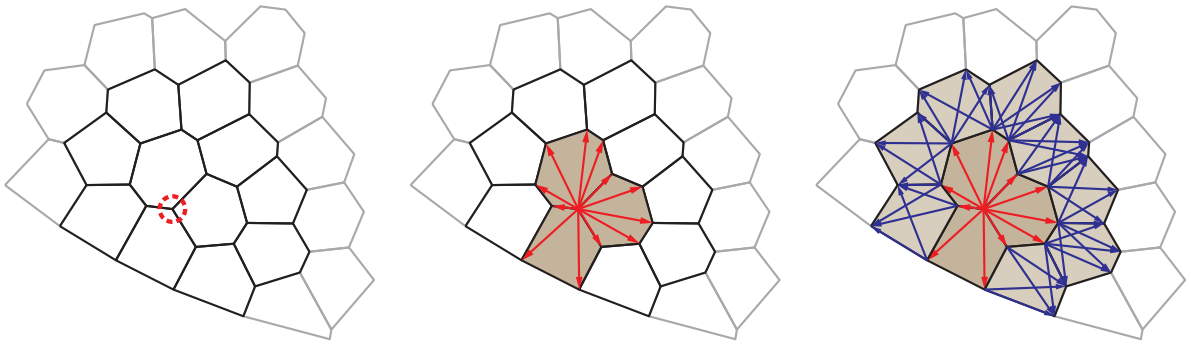
Figure 4.4: Member number growth using the GRAND ground structure generation algorithm. (a) Member generation for the polygonal element base mesh shown in Figure 4.3(a). (b) Member generation for a structured and orthogonal mesh with 30×10 square elements.

$\mathbf{A}_1$, and is defined as follows:

$$[\mathbf{A}_1]_{p,q} = \begin{cases} 1 \text{ or } \texttt{true} & \text{if nodes } p, q \text{ share an element} \\ 0 \text{ or } \texttt{false} & \text{otherwise or if } p = q \end{cases} \tag{4.7}$$

The second level connectivity is simply $\mathbf{A}_2 = \mathbf{A}_1\mathbf{A}_1$. However, it should be noted that this matrix is likely to have entries > 1 and a non–zero diagonal (see the example in Figure 4.5). Thus, the diagonal is set to zero and the matrix is again converted to *logical*: *true* or 1 for any value larger than 1, and *false* or 0 otherwise. The nodal connectivity matrix for some level $n > 1$ is then:

$$[\mathbf{A}_n]_{p,q} = \begin{cases} 0 \text{ or } \texttt{false} & \text{if } p = q \\ 1 \text{ or } \texttt{true} & \text{if } [\mathbf{A}_1^n]_{p,q} > 0 \\ 0 \text{ or } \texttt{false} & \text{otherwise} \end{cases} \tag{4.8}$$



(a) (b) (c)

Figure 4.5: Connectivity matrix calculation. (a) Base mesh and starting node. (b) Level 1 connectivity obtained from $\mathbf{A}_1$. (c) Level 2 connectivity. Note that the entries of $\mathbf{A}_2 = (\mathbf{A}_1)^2$ are typically > 1 due to the existence of more than one path to the new set of nodes.

4.2.3 Collinearity check

The following assumptions are made in the collinearity check:

1. New bars added at level n are deleted if found collinear with bars from previous levels.

2. New bars added at level n are assumed not to be collinear between them.

The first assumption is logical if we consider that new bars are *longer* than those from previous levels. The second assumption may be violated if elements in the base mesh are not strictly convex, or in domain shapes that curl; where a new level may have two (or more) collinear nodes viewed from the starting node (Figure 4.6). The new (candidate) bars for level n can be obtained as:

$$\mathbf{G}_n = \mathbf{A}_n - \mathbf{A}_{n-1} \,, \tag{4.9}$$

with $\mathbf{G}_1 = \mathbf{A}_1$. The non–zero entries in $\mathbf{G}_n$ that will be included in the ground structure are those that have no angle close to zero with previously accepted bars. The directional

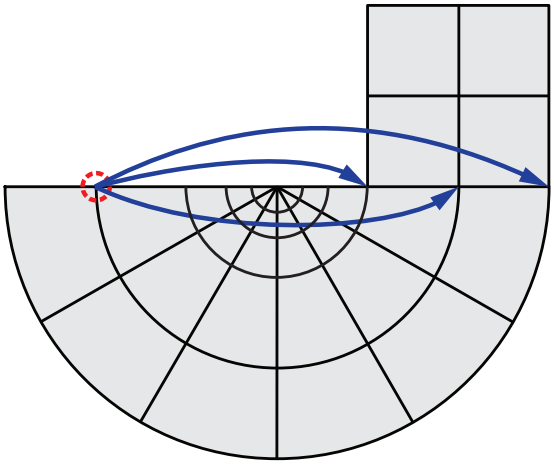


Figure 4.6: Domain that curls: The highlighted node will generate collinear members at level 6. The generation algorithm will reach these three nodes at the same time, and collinearity between them will not be checked.

cosines vector $\hat{\mathbf{d}}_{p,q}$ from node p to node q is:

$$\begin{aligned} \mathbf{d}_{p,q} &= \texttt{NODE}_{q,:} - \texttt{NODE}_{p,:} \\ \hat{\mathbf{d}}_{p,q} &= \frac{\mathbf{d}_{p,q}}{\|\mathbf{d}_{p,q}\|} \,, \end{aligned} \tag{4.10}$$

where $\texttt{NODE}_{i,:}$ stands for row i of the nodal coordinates array (i.e. the coordinates x and y of node i), as defined in Table 4.1. The angle between two directional cosines vectors is:

$$\cos\left(\angle qpr\right) = \hat{\mathbf{d}}_{p,q} \cdot \hat{\mathbf{d}}_{p,r} \tag{4.11}$$

Assume that m new candidate bars originating from a specific node i, have to be checked against previously accepted n bars from the same node. The directional vectors of the new (candidate) bars can be grouped into $\mathbf{D}_{new}$ of size $m \times 2$, and the previously accepted bars into $\mathbf{D}_{old}$ of size $n \times 2$. The dot product of new (candidate) and old bars can be (efficiently) computed by:

$$\mathbf{C} = \mathbf{D}_{old}\mathbf{D}^{\mathbf{T}}{}_{new} \,, \tag{4.12}$$

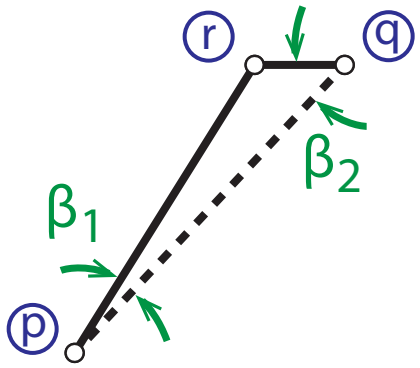


Figure 4.7: Collinearity test between three bars. The long bar (dashed line) between nodes p and q is candidate for deletion.

where a new (candidate) bar j is found to be collinear with a previously accepted bar if any entry in column j is equal to 1. In reality, a collinearity tolerance $ColTol \lessapprox 1$ is used instead, and a bar j is removed if:

$$\mathbf{C}_{i,j} > ColTol \qquad \forall\, i = 1 \ldots n \tag{4.13}$$

The bars that passes the collinearity test are then appended to the ground structure.

The currently accepted bars at a level are stored in matrix $\mathbf{H}$, with:

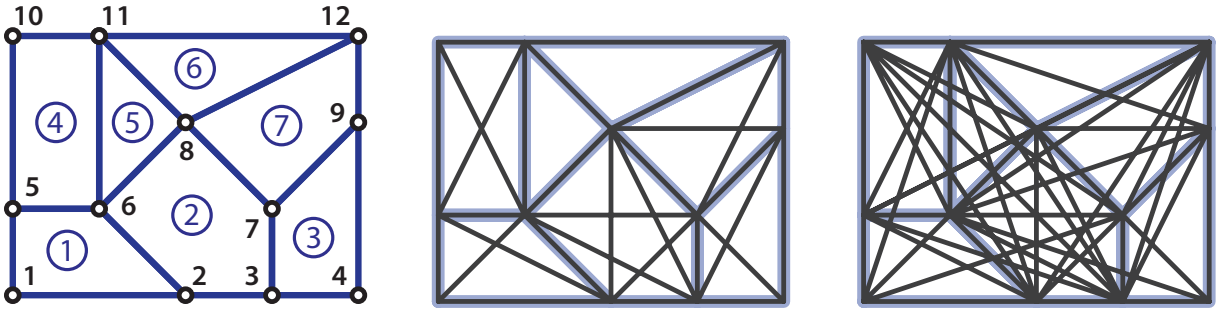
$$\mathbf{H}_{p,q} = \begin{cases} 1 \text{ or } \texttt{true} & \text{if there } \exists \text{ member } \overline{pq} \\ 0 \text{ or } \texttt{false} & \text{otherwise} \end{cases} \tag{4.14}$$

Matrix $\mathbf{H}$ looses symmetry when a bar has one angle below $ColTol$ and one above; in the example of Figure 4.7 this means $\cos(\beta_2) < ColTol < \cos(\beta_1)$, and thus $\mathbf{H}_{p,q} = 0$ but $\mathbf{H}_{q,p} = 1$. If the member has one angle that pass the $ColTol$ requirement, it will be spared from deletion by forcing matrix $\mathbf{H}$ to be symmetric again:

$$\mathbf{H}^{\star} = \mathbf{H} + \mathbf{H}^{\mathbf{T}}\,, \tag{4.15}$$

after each level calculation. The entries $\mathbf{H}^\star > 0$ contain the bars of the *current* ground structure to be used in the next level iteration if any.

The generation algorithm stops once it reached the specified connectivity level Lvl, or earlier if $\mathbf{G}_n$ has no entries (all zero or *false* matrix). Once the ground structure has been generated, only bars linking nodes p and q with $p < q$ are returned (i.e. the upper triangular form of $\mathbf{H}^\star$). This prevents duplicate bars from appearing in the ground structure.



(a) (b) (c)

Figure 4.8: Ground structure generation example. (a) Sample base mesh with 7 elements and 12 nodes. (b) Resulting ground structure for a level 1 connectivity. (c) Resulting ground structure for a level 2 connectivity.

The sample base mesh in Figure 4.8(a) results in the matrices $\mathbf{A}_1 = \mathbf{G}_1$ and $\mathbf{G}_2$ detailed in Equations (4.16) and (4.17).

$$\mathbf{A}_1 = \mathbf{G}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ & & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ & & & & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ & & & & & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ & & & & & & 0 & 1 & 1 & 0 & 0 & 1 \\ & & & & & & & 0 & 1 & 0 & 1 & 1 \\ & & & & & & & & 0 & 0 & 0 & 1 \\ & & & \text{symm} & & & & & & 0 & 1 & 0 \\ & & & & & & & & & & 0 & 1 \\ & & & & & & & & & & & 0 \end{bmatrix} \tag{4.16}$$

$$\mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ & & & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ & & & & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ & & & & & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & & & 0 & 0 & 0 & 1 & 1 & 0 \\ & & & & & & & 0 & 0 & 1 & 0 & 0 \\ & & & & & & & & 0 & 0 & 1 & 0 \\ & & \text{symm} & & & & & & & 0 & 0 & 1 \\ & & & & & & & & & & 0 & 0 \\ & & & & & & & & & & & 0 \end{bmatrix} \tag{4.17}$$

The members from level 1 are always included in the ground structure (no collinearity check), resulting in the ground structure in Figure 4.8(b). The candidate bars at level 2 are tested against the previously accepted bars at level 1, and thus not all members in $\mathbf{G}_2$ will be retained. The accepted bars for levels 1 and 2 are detailed in matrix $\mathbf{H}^\star$ as follows:

$$\mathbf{H}^\star = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & & & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ & & & & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ & & & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ & & & & & & 0 & 1 & 1 & 1 & 0 & 1 \\ & & & & & & & 0 & 1 & 1 & 1 & 1 \\ & & & & & & & & 0 & 0 & 1 & 1 \\ & & \text{symm} & & & & & & & 0 & 1 & 0 \\ & & & & & & & & & & 0 & 1 \\ & & & & & & & & & & & 0 \end{bmatrix} \tag{4.18}$$

Note, for example, that node 1 does not link to nodes 3, 8 or 10 because these (level 2) bars are collinear with members from previous levels. The output of the ground structure algorithm for the base mesh in Figure 4.8(a), for a level 2 connectivity with no collinear bars

results in the following 48 members:

$$\texttt{BARS} = \left[\begin{array}{cc} 1 & 2 \\ 1 & 5 \end{array}\right] \left[\begin{array}{cc} 1 & 6 \\ 1 & 7 \end{array}\right] \left[\begin{array}{cc} 1 & 11 \\ 2 & 3 \end{array}\right] \left[\begin{array}{cc} 2 & 5 \\ 2 & 6 \end{array}\right] \left[\begin{array}{cc} 2 & 7 \\ 2 & 8 \end{array}\right] \left[\begin{array}{cc} 2 & 10 \\ 2 & 11 \end{array}\right] \left[\begin{array}{cc} 2 & 12 \\ \vdots & \vdots \end{array}\right] \left[\begin{array}{cc} \vdots & \vdots \\ 3 & 4 \end{array}\right] \left[\begin{array}{cc} 3 & 5 \\ 3 & 6 \end{array}\right] \left[\begin{array}{cc} 3 & 7 \\ 3 & 8 \end{array}\right] \left[\begin{array}{cc} 3 & 9 \\ 3 & 10 \end{array}\right] \left[\begin{array}{cc} 3 & 11 \\ 3 & 12 \end{array}\right] \left[\begin{array}{cc} 4 & 6 \\ 4 & 7 \end{array}\right] \left[\begin{array}{cc} \vdots & \vdots \\ \vdots & \vdots \end{array}\right] \left[\begin{array}{cc} \vdots & \vdots \\ 4 & 9 \end{array}\right] \left[\begin{array}{cc} 5 & 6 \\ 5 & 8 \end{array}\right] \left[\begin{array}{cc} 5 & 10 \\ 5 & 11 \end{array}\right] \left[\begin{array}{cc} 5 & 12 \\ 6 & 7 \end{array}\right] \left[\begin{array}{cc} 6 & 8 \\ 6 & 9 \end{array}\right] \left[\begin{array}{cc} 6 & 10 \\ 6 & 11 \end{array}\right] \left[\begin{array}{cc} \vdots & \vdots \\ \vdots & \vdots \end{array}\right] \left[\begin{array}{cc} \vdots & \vdots \\ 6 & 12 \end{array}\right] \left[\begin{array}{cc} 7 & 8 \\ 7 & 9 \end{array}\right] \left[\begin{array}{cc} 7 & 10 \\ 7 & 12 \end{array}\right] \left[\begin{array}{cc} 8 & 9 \\ 8 & 10 \end{array}\right] \left[\begin{array}{cc} 8 & 11 \\ 8 & 12 \end{array}\right] \left[\begin{array}{cc} 9 & 11 \\ 9 & 12 \end{array}\right] \left[\begin{array}{cc} 10 & 11 \\ 11 & 12 \end{array}\right] \tag{4.19}$$

The bar connectivity information in Equation (4.19) results in the ground structure in Figure 4.8(c).

4.2.4 Restriction zones

The restriction zone idea is inspired by known collision detection algorithms used in video–games and in computational geometry (Ericson, 2004).

The domain may have concave regions or holes where no bar should be present. To prevent the ground structure generation algorithm from laying out members in these regions, the user defines *restriction zones* (i.e. hitboxes): geometric entities that no bar should intersect. The current implementation provides the following restriction primitives:

- rectangle
- circle
- segment (line)
- convex polygon

The restriction primitives return a *true* or 1 if a member comes in contact with them. These primitives can be combined using logical operators to create complicated regions or boundaries, and the user can easily implement new collision primitives in GRAND. Union, intersection and substraction of primitives are all possible using these logical operators. The restriction zone for the cantilever with circular support is illustrated in Figure 4.9 as an example. Candidate bars for levels > 1 are tested against the restriction zones. Level 1, however, is not tested because it is assumed that the base input mesh is completely contained within the feasible domain.

If a node in the domain touches the boundary of a restriction primitive, then all members originating from that node will be flagged from removal (except for level 1 that does not get tested). To avoid unexpected removals, a small reduction of the restriction zone primitives is advised. This setback (or margin) is applied in the form of a small tolerance *tol* as explained in Figure 4.10. The setback *tol* is specified by the user, and should be relative to the scale of the problem.

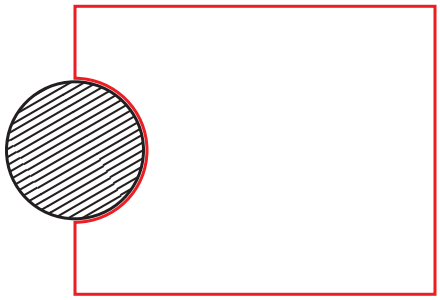


Figure 4.9: Restriction zones for the cantilever with circular support detailed in Figure 4.1.

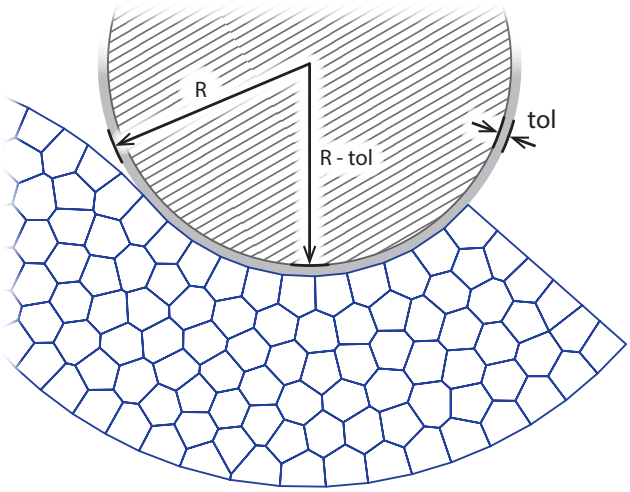


Figure 4.10: Restriction zone setback to prevent nodes in the domain to come in contact with the restriction zones. The setback is a margin of size *tol*, relatively small compared to the scale of the domain.

4.2.5 Linear program input/output

The matrix $\mathbf{B^T}$ in Equation 4.6 is assembled from the directional cosines $\hat{\mathbf{d}}$, and the member length vector denoted as $\mathbf{l} = \|\mathbf{d}_i\|$. The nodal force vector $\mathbf{f}$ is generated with the information supplied in the `LOAD` array.

With no loss of generality, the implementation assumes $\sigma_T = 1$. The case of $\sigma_T \neq 1$ is scaled from the results obtained with the previous assumption, as follows:

$$\begin{aligned} \mathbf{a}_{\sigma_T \neq 1} &= \mathbf{a}/\sigma_T \\ V_{\sigma_T \neq 1} &= \left(V_{\sigma_T = 1}\right)/\sigma_T \end{aligned} \tag{4.20}$$

4.2.6 Plotting scheme

Solutions with a large number of members may have plotting issues. To prevent this, only the members with cross–sectional areas $a_i > (\mathit{Cutoff})\,(a_{max})$ are plotted: this may cause a truss cord to abruptly end mid–air in the resulting figure, which is not accurate, but is merely a plotting artifact. Members are plotted with lines of varying thicknesses. The member

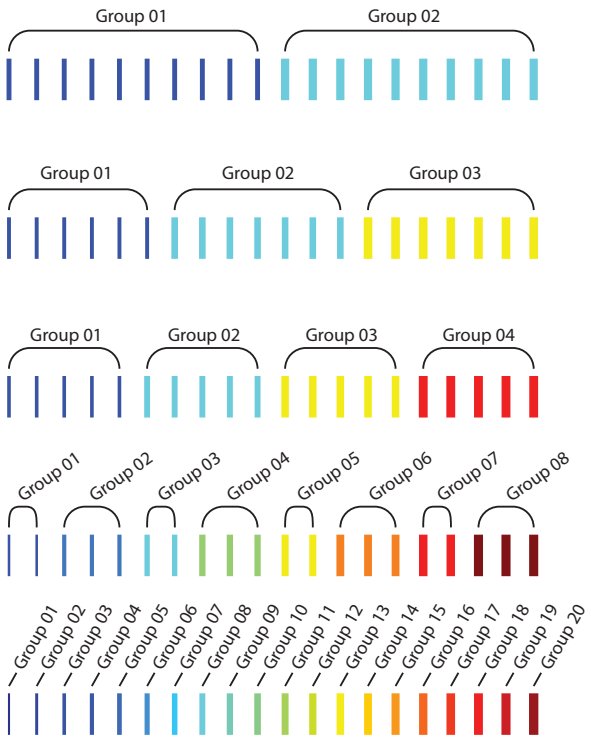


Figure 4.11: Plotting of 20 members with cross–sectional areas $a_i = i/20$ for $i = 1 \ldots 20$ using 2, 3, 4, 8 and 20 plotting groups.

thickness is proportional to the square root of its cross–sectional area. In other words:

$$\text{LineWidth}_i = c\sqrt{a_i}\,, \tag{4.21}$$

where c is a scaling constant. The scaling constant is chosen such that the thickest member is 5 *points* width.

To reduce the plot function calls, members of similar areas are grouped and plotted together with average visual characteristics (thickness and color), in a single function call. The user can specify the number of groups used in this plot grouping technique. Provided that sufficient groups are specified, the error introduced is small (Figure 4.11).

Table 4.2: Square cantilever beam comparison.

30×10 mesh with Level 10 connectivity	GRAND code	Structured implementation*
DOFs	660	660
Bars	19,632	19,632
Volume	13.6857	13.6857

* The structured implementation refers to Sokół (2011)

4.3 Examples and verification

The following problems were selected to showcase features and issues in GRAND. Unless stated otherwise, the stress limit ratio is $\kappa = 1.0$ with $\sigma_T = 1$, the collinear tolerance is $ColTol = 0.999999$, and the plotting cutoff is $Cutoff = 0.002$, using 50 plotting groups; similar members are grouped and plotted together.

4.3.1 Structured square cantilever

The ground structure generation algorithm for unstructured meshes (detailed in this chapter), must generate the same ground structure as an implementation of the method for structured orthogonal meshes. A rectangular cantilever beam clamped at the left side and loaded at the right by a vertical force applied at the middle–central point is used for comparison: the domain size is 3×1, discretized using 30×10 elements with a unit downward vertical load and $Cutoff = 10^{-6}$. Table 4.2 and Figure 4.12 summarize the number of truss elements, degrees–of–freedom (DOFs), optimal volume and resulting structure for both methods: the solution using GRAND is exactly the same as the reference implementation (Sokół, 2011).

4.3.2 Cantilever with circular support

This problem is also known as the *Michell's cantilever* (Figure 4.1). Due to the geometry of the domain, this problem requires an unstructured mesh to be modeled accurately. This problem has a known analytical solution (Michell, 1904; Graczykowski and Lewiński, 2005)

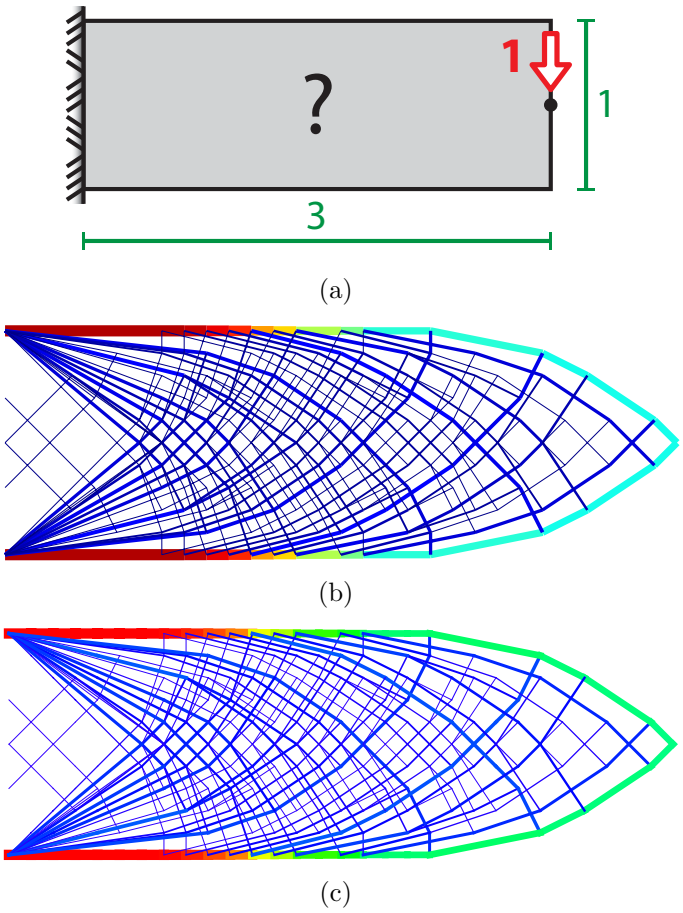


Figure 4.12: Cantilever loaded at the mid–tip (a) Domain definition, discretized with 30×10 elements and level 10 connectivity (b) Solution from GRAND (c) Solution from a structured ground structure implementation (Sokół, 2011).

if the problem height H is sufficiently large to allow the full solution to develop:

$$V_{opt} = P \log\left(\frac{r}{R}\right)\left[\frac{1}{\sigma_T} + \frac{1}{\sigma_C}\right] \tag{4.22}$$

Given the following parameters for the problem; $r = 1$, $R = 5$, $H = 4$ and $P = 1$, the optimal volume is $V_{opt} = 16.0944$. The convergence with refinement to the analytical solution is shown in Figure 4.13(a), with the values taken from Table 4.3. One of these solutions is shown in Figure 4.13(b) as an example.

Table 4.3: Cantilever with circular support. Problem parameters: $r = 1$, $R = 5$, $H = 4$ and $P = 1$.

N_e	N_n	Lvl	N_b	Volume V
300	593	1	3,264	16.9824
625	1,239	2	20,447	16.2777
1,200	2,369	3	78,807	16.1834
2,800	5,527	4	313,830	16.1396
5,000	9,889	5	851,511	16.1234
10,500	20,761	6	2,548,545	16.1140

4.3.3 Hook problem

The hook problem (Figure 4.14(a)) initially introduced by Talischi et al. (2012b) has no analytical solution, yet the complexity of the domain makes it a good example to showcase the capabilities of the proposed method. The domain has a restriction zone composed of four primitives as in Figure 4.14(b): three circles and one segment. For comparison, the solution from a polygonal density–based method (Talischi et al., 2012b) is also provided: there is a qualitative agreement of the solutions from both methods (Figures 4.14(c) and 4.14(d)).

4.3.4 Serpentine cantilever

The serpentine cantilever, introduced in Talischi et al. (2012b), is meshed using 600 polygonal elements and 30 Lloyd's iterations (Talischi et al., 2012a). The ground structure is then generated for a level 5 connectivity, using the restriction zone illustrated in Figure 4.15(b). The restriction zone is the union of 2 circle restriction primitives. The domain, restriction zone, base mesh and optimized ground structure are illustrated in Figure 4.15.

The solution, despite it being coarse, is rich in fan and involute structures. In addition, the members tend to cross at perpendicular angles. These are all signs of an optimal solution, and the quality and quantity of these increase with mesh refinement.

4.3.5 Messerschmitt–Bölkow–Blohm (MBB) beam

The MBB beam is a typical problem in topology optimization (Bendsøe and Sigmund, 2003; Lewiński et al., 1994a). The domain size is $L_x \times L_y = 6 \times 1$, and is meshed with a structural and orthogonal arrangement of 120×20 quads (using an internal GRAND function that generates structured–orthogonal meshes). The ground structure is constructed for a level 6 connectivity with no restriction zone since the domain is convex. The domain, base mesh, optimized ground structure and reference analytical solution are illustrated in Figure 4.16. As expected, the (numerical) solution obtained with GRAND agrees with the known analytical solution.

4.3.6 Flower problem

The flower problem is a donut–shaped domain loaded tangentially in the exterior at 5 equal angle locations. The domain is fully supported at the interior radius (Figure 4.17(a)). The mesh is structured, however, this domain does require a restriction zone for the donut hole (Figure 4.17(b)).

The optimal solution to this problem is composed by 5 *Michell's cantilevers* (cantilever with circular support as shown in Section 4.3.2). These are distributed at equidistant angles around the interior circle, with each cantilever supporting one of the loads applied in the exterior circle. The analytical solution to this problem is:

$$V_{opt} = 5PR \log\left(\frac{R}{r}\right)\left[\frac{1}{\sigma_T} + \frac{1}{\sigma_C}\right] = 13.8629 \tag{4.23}$$

The volume obtained using the base mesh in Figure 4.17(c) is $V_{grand} = 13.9674$. This (numerically computed) volume is less than 1% higher than the known optimal.

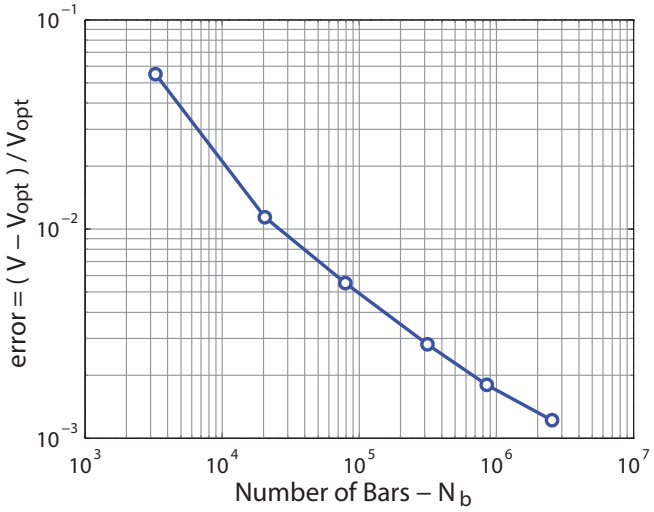
This example illustrates the capability of GRAND for generating a biologically inspired structure, which displays the patterns of a flower, as shown by Figures 4.17(d) and 4.17(e).

4.4 Conclusions

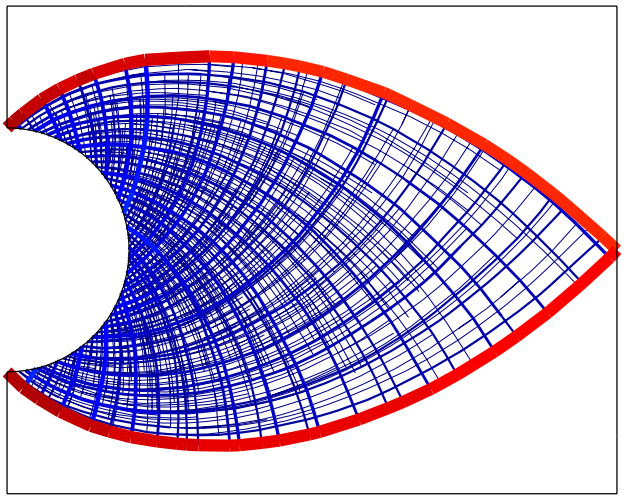
The ground structure method provides insight into the optimal solution for a given problem. This information can be used to further refine the solution, and in some cases, to obtain an analytical solution (or benchmark). The present method and implementation extends the ground structure method to domains of any shape with a simple and efficient technique.

To guarantee quality solutions, the ground structure should have linear independence among truss members (no collinearity), and no two members connecting the same nodes (repeated members). Concave domains and holes are addressed by defining *restriction zones*, that are constructed with collision primitives common in the video game and rendering industries. These zones flag colliding members so that they can be removed when generating the ground structure. The member generation, collinearity check, restriction calculation and member removal are translated into linear algebra operations that can be efficiently calculated in modern linear algebra systems.

A freely–available MATLAB implementation is provided to encourage future research in the field and an in–depth understanding of the method. This implementation was created with a balance of performance and readability in mind, and can be further improved for speed and flexibility if required. In particular, methods that adaptively generate, modify and/or reduce the ground structure have been successful at achieving greater level of detail, while maintaining a reasonable computational cost (Gilbert and Tyas, 2003; Rozvany and Sokół, 2013).



(a)



(b)

Figure 4.13: Cantilever with circular support. (a) Convergence with ground structure refinement. (b) Solution obtained for $N_b = 851,511$, generated from a non–symmetric (unstructured) polygonal mesh with $N_e = 5,000$, $N_n = 9,889$ and $Lvl = 5$.

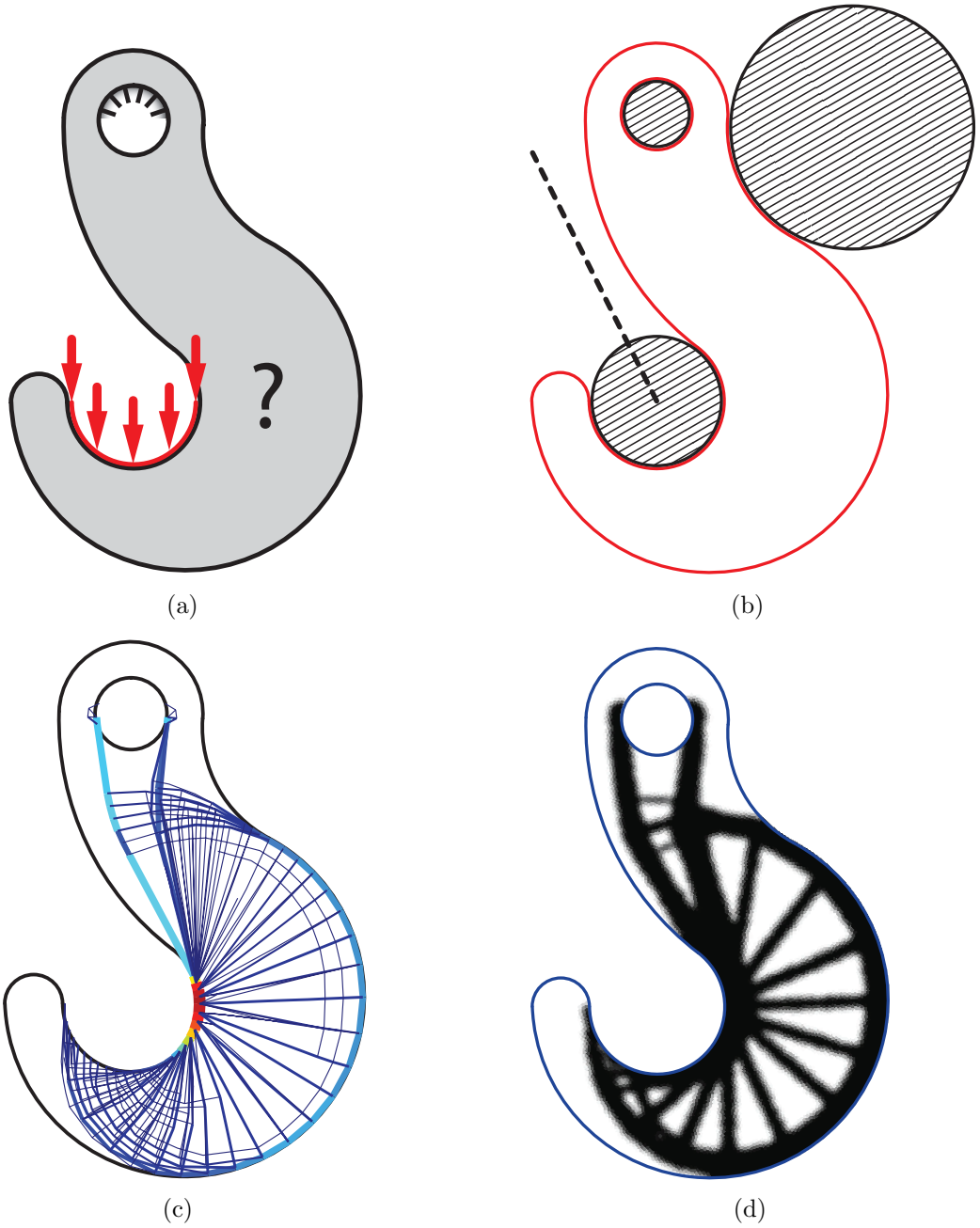


Figure 4.14: Hook problem: (a) Domain, loading and boundary conditions. (b) Restriction zone composed of three circles and one segment. (c) Solution obtained from GRAND with $N_b = 72,589$ using an externally generated mesh and level 10 connectivity. (d) Solution from a density method with $N_e = 10,000$ (continuum polygonal elements) for comparison.

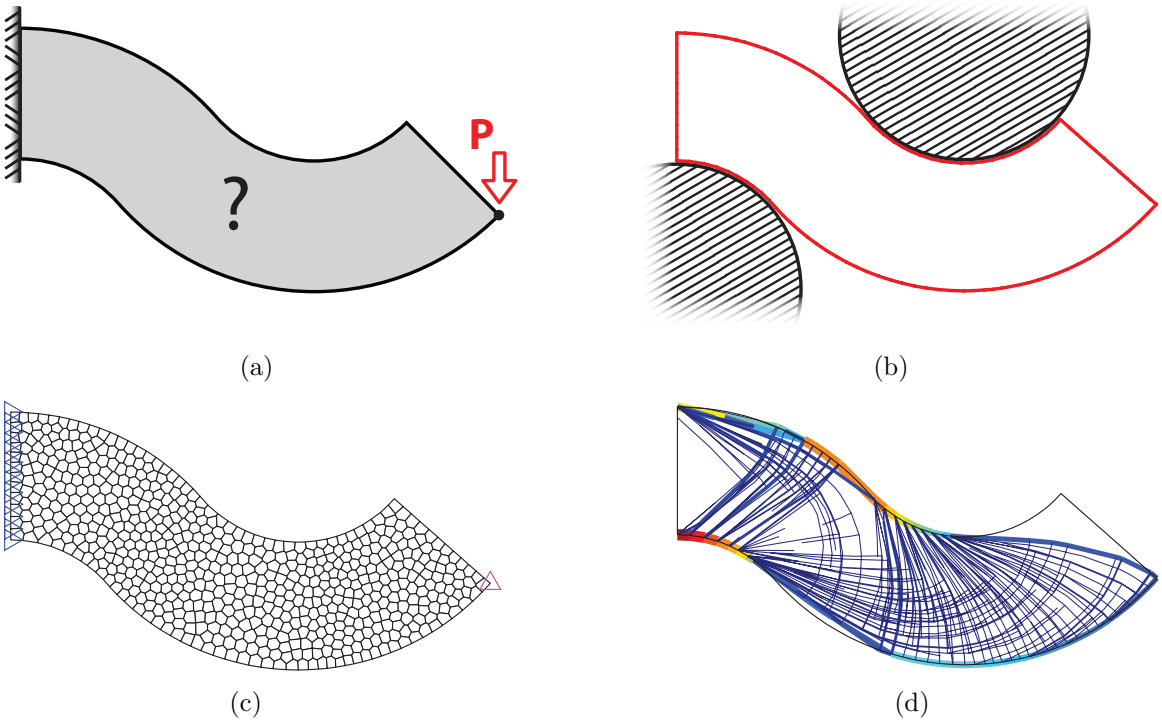


Figure 4.15: Serpentine cantilever problem: (a) Domain, loading and boundary conditions. (b) Restriction zone composed of two circles. (c) Serpentine domain discretized using polygonal elements (Talischi et al., 2012a): $N_e = 600$ and $N_n = 1,192$. Nodes with prescribed displacements and forces are highlighted with a blue $\triangleright$ and a magenta $\triangle$ respectively. (d) Solution obtained from GRAND with $lvl = 5$ and $N_b = 1,192$.

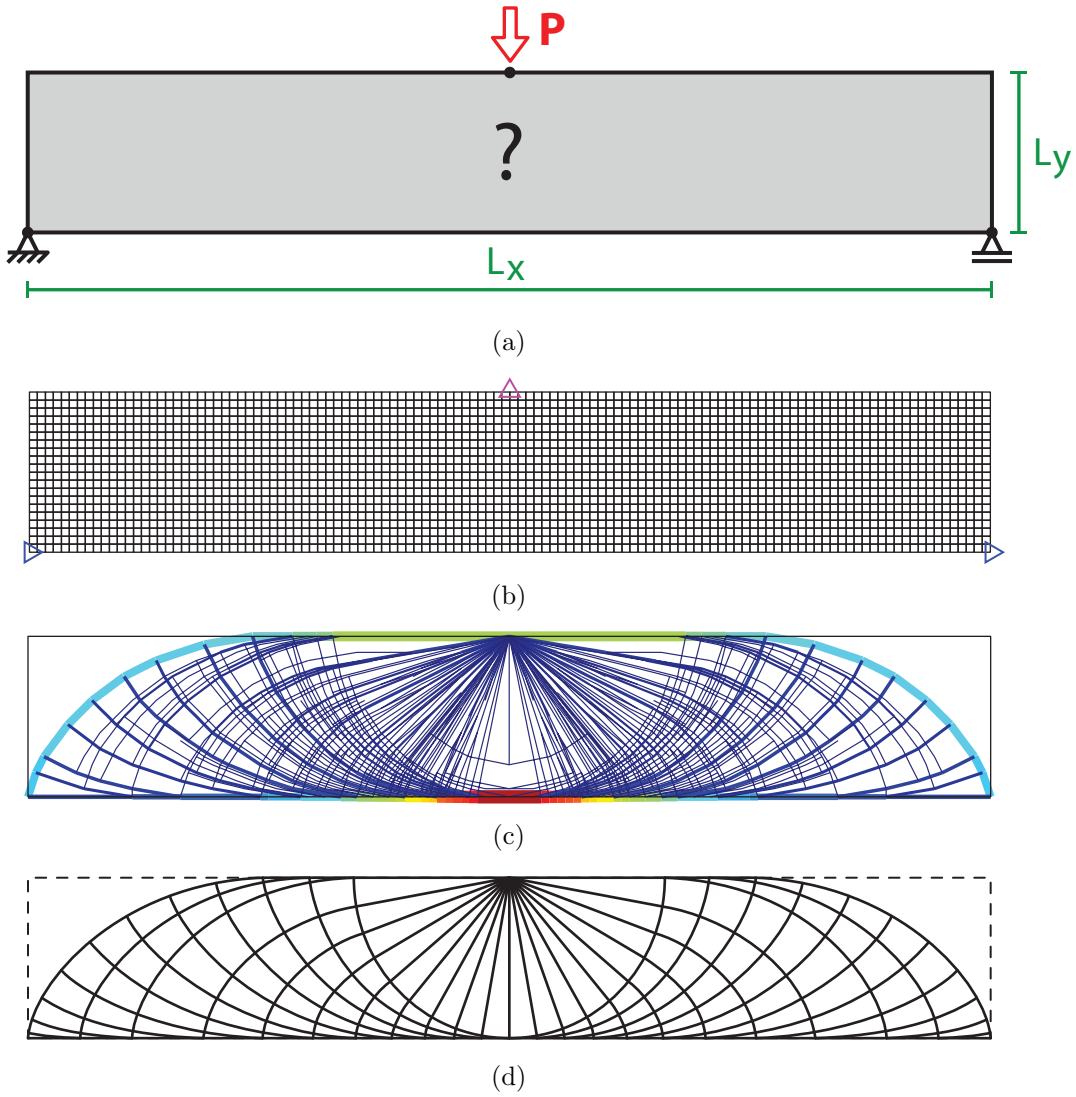


Figure 4.16: Messerschmitt–Bölkow–Blohm (MBB) beam problem with aspect ratio $L_x : L_y = 6 : 1$. (a) Domain, loading and boundary conditions. (b) MBB domain discretized with a regular and orthogonal base mesh in GRAND: $N_e = 120 \times 20 = 2,400$ and $N_n = 2,541$. Nodes with prescribed displacements and forces are highlighted with a blue $\triangleright$ and a magenta $\triangle$ respectively. (c) Optimized ground structure for the MBB domain: $lvl = 6$ and $N_b = 101,548$. (d) Analytical solution adapted from Lewiński et al. (1994a).

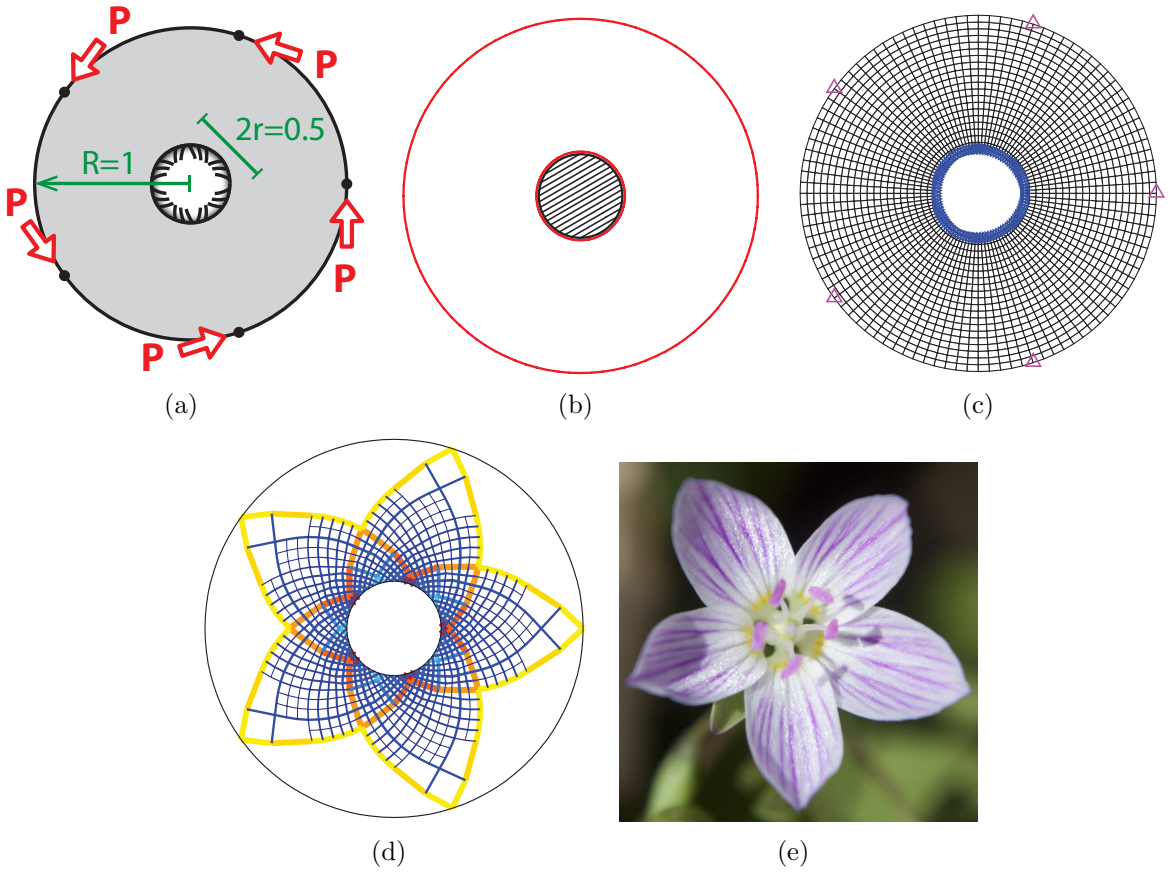


Figure 4.17: Flower problem (donut–shaped domain) loaded tangentially at 5 locations on the outer radius. (a) Domain, loading and boundary conditions. (b) Restriction zone for the donut–shaped domain. (c) Mesh and boundary conditions are loaded from an externally generated file: $N_e = 2,000$ and $N_n = 2,100$. (d) Optimized ground structure for the flower domain: $lvl = 4$ and $N_b = 69,400$. (e) Photo of *Claytonia caroliniana* [© Nathan Masters | Masters Imaging].

Chapter 5

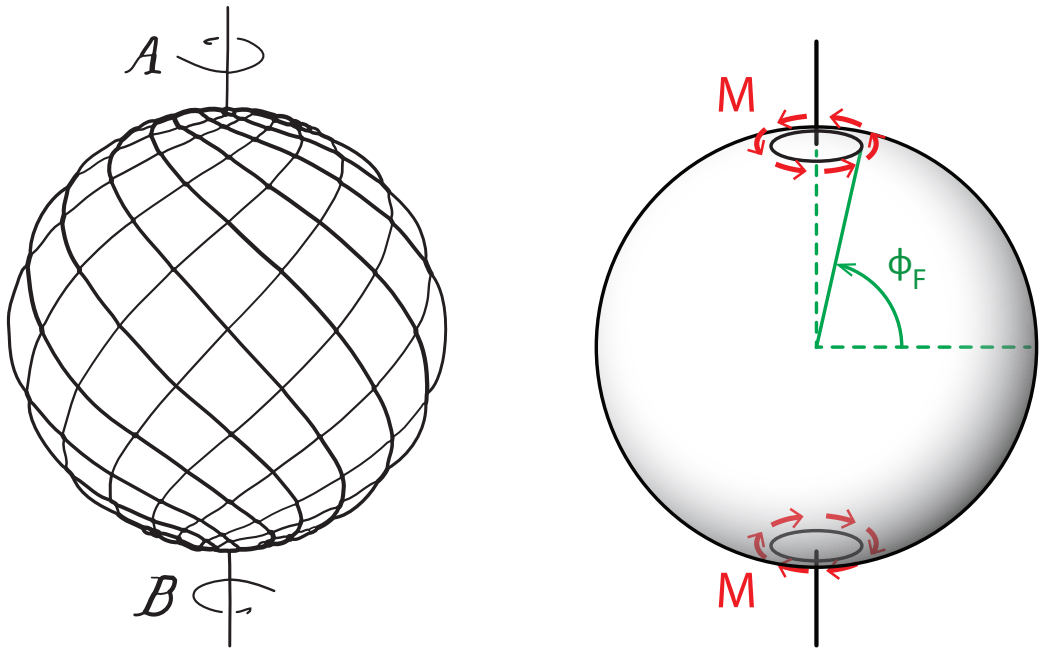
Unstructured ground structure method in 3D

The search for optimal structures was reinvigorated with Michell's work (Michell, 1904). His work on minimum weight structures determined the best possible configuration for a variety of structural problems (also known as Michell's solutions). Later, Hemp (1973) reanalyzed these and other examples in more detail. Michell's work focused mostly on two–dimensional structures, and only one three–dimensional problem was addressed. This problem is known as the Michell's sphere (or torsion ball), and is the optimal (analytical) structure to transfer a moment couple. This truss–like solution is a capped sphere structure with a grid of orthogonal members on its surface (Figure 5.1(a)). The optimal volume for this problem was given by Michell (1904), but a detailed derivation was not given. The solution for the torsion ball was later re–derived (Hemp, 1973; Lewiński, 2004) and confirmed[1]. The optimal volume of the torsion ball is:

$$V_{opt} = 2M \log\left(\tan\left\{\frac{\pi}{4} + \frac{\phi_F}{2}\right\}\right)\left[\frac{1}{\sigma_T} + \frac{1}{\sigma_C}\right] , \tag{5.1}$$

where ϕ_F is the *latitude* of the small circles where the moment is applied (Figure 5.1(b)). It should be noted that the optimal volume is independent of the sphere's radius r. Nonetheless, the domain must be large enough to allow the solution's sphere with radius r to develop. It

[1]Michell's formula matches the subsequent work provided that the quantity L in Michell (1904) is taken to be equal to Mr. Whether Michell meant this to be the real meaning of the quantity L, or not, is unclear.



(a) (b)

Figure 5.1: Optimal (analytical) structure to transfer a moment couple. (a) Distribution of the members according to Michell (1904). (b) Illustration of the latitude ϕ_F, which defines the small circles where the moment couples are applied.

should be noted that this is the optimal truss–like solution: if the solution sought is of the continuum type, then the solution becomes a hollow spherical shell with varying thickness. The inclusion of additional constraints in the continuum solution can drive it towards the aforementioned truss–like structure (Aage et al., 2014).

In later years, the topic of optimal three–dimensional structures was revisited, and the solution to a number of additional problems was given (Lewiński, 2004). The ground structure method (Dorn et al., 1964; Hemp, 1973) was extended to three–dimensional problems (Gerdes, 1994; Smith, 1998; Gilbert et al., 2005; Tyas et al., 2006) with promising results. Most of these works, with the exception of Smith (1998), tackle problems in rectangular domains. Other numerical methods have also been utilized to address the 3D layout optimization problem with mixed results (Zhou and Li, 2005; Dewhurst and Taggart, 2009). However, those methods' ability to approximate the analytical solution is usually less than

that of the ground structure method.

Real structures are three–dimensional, and thus the requirement of a *true* three–dimensional analysis is obvious. Therefore, the unstructured ground structure method presented in the previous chapter is extended to 3D space. The implementation of the method is named GRAND3 (GRound structure ANalysis and Design in 3D), and is a direct extension of the 2D implementation (GRAND) described in Chapter 4 and in Zegard and Paulino (2014a). The approach used in GRAND3 makes it easy to model, analyze and optimize, three–dimensional domains of almost any shape (concave and with the possibility of holes) using the ground structure method. The complete source code for GRAND3 is available in Appendix B.

5.1 Plastic analysis formulation in 3D

The plastic formulation (Dorn et al., 1964; Hemp, 1973) is extended to three–dimensional space. The dimensions of the matrices and vectors involved are modified to accommodate the third dimension. Starting from Equation (4.6), the formulation in three–dimensional space (with the sizes of matrices and vectors indicated) is:

$$\begin{aligned} \min_{\mathbf{s}^+,\mathbf{s}^-} \quad & V^\star = \frac{V}{\sigma_T} = \underset{1\times 2N_b}{\begin{Bmatrix} \mathbf{l}^T & \kappa\,\mathbf{l}^T \end{Bmatrix}} \underset{2N_b\times 1}{\begin{Bmatrix} \mathbf{s}^+ \\ \mathbf{s}^- \end{Bmatrix}} \\ \text{s.t.} \quad & \underset{N_{dof}\times 2N_b}{\begin{bmatrix} \mathbf{B^T} & -\mathbf{B^T} \end{bmatrix}} \underset{2N_b\times 1}{\begin{Bmatrix} \mathbf{s}^+ \\ \mathbf{s}^- \end{Bmatrix}} = \underset{N_{dof}\times 1}{\mathbf{f}} \\ & s_i^+, s_i^- \geq 0 \end{aligned} \tag{5.2}$$

The number of degrees–of–freedom (DOFs) is $N_{dof} = 3N_n - N_{sup}$ (previously in two–dimensions $N_{dof} = 2N_n - N_{sup}$). Equation (5.2) highlights the sizes of the variables involved: The objective suffers no change, but the equilibrium constraints grow by (approximately) a factor of 1.5. The computational complexity depends mostly on the number of variables

n in the linear program; for the particular case of the plastic formulation $n = 2N_b$, with N_b being the number of members in the ground structure. The implementation, written in MATLAB, uses the linear programming routine within MATLAB's *Optimization Toolbox*, itself being a modified version of the LIPSOL library (Zhang, 1998). Linear programming algorithms typically exhibit computational complexities in the range of $\mathcal{O}(n^4)$ to $\mathcal{O}(n^3)$ and lower (Anstreicher, 1999; Wright, 2004). For a given number of members N_b in the ground structure, the runtime should remain mostly unaffected by extending the method to three–dimensions. In practice, however, the three–dimensional ground structure method is slower than its two–dimensional counterpart. This will depend on the specific variation of the interior–point method used, in this case LIPSOL (Zhang, 1998). In addition, problems with solutions that lie on a facet (or edge) of the feasible domain are computationally more difficult to obtain.

5.2 Implementation

The implementation is an extension of the two–dimensional GRAND (refer to Chapter 4 and Zegard and Paulino (2014a)). The concept and algorithms remain mostly intact, with a few exceptions that are outlined and detailed here.

5.2.1 Domain definition — Base mesh

The ground structure generation algorithm requires a base mesh. The nodal connectivity information (required to generate the ground structure) is obtained from this mesh, but no edge or facet information is used. The method assumes that all elements in the base mesh are strictly convex. Thus, it is safe to connect all nodes within an element for a level 1 connectivity (trivial extension of the conclusions in Section 4.2.3). Given these assumptions, the method can handle any type of convex polytope elements in the base mesh. This, however, proves to be challenging for plotting routines since no edge and facet information are

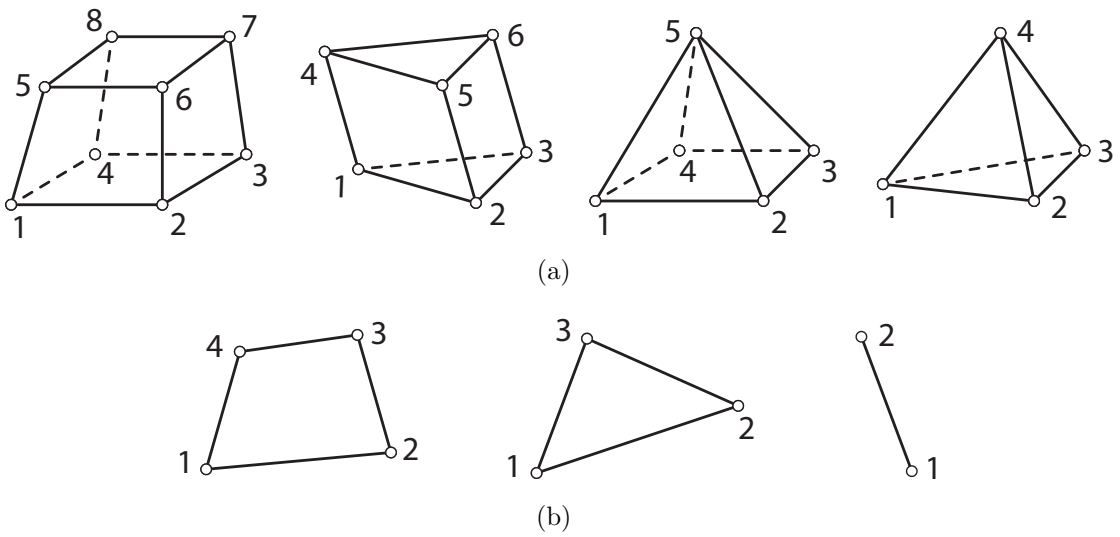


Figure 5.2: Elements supported by GRAND3 and their corresponding node numbering scheme. (a) Volumetric elements: Hexahedron (8-nodes), Prism (6-nodes), Pyramid (5-nodes) and Tetrahedron (4-nodes). (b) Surface elements: Quadrangle (4-nodes), Triangle (3-nodes) and Segment (2-nodes).

provided. In addition, the maturity of 3D polytope meshing algorithms (Barber et al., 1996; Herceg et al., 2013; Rycroft, 2014) is lagging behind that of more traditional meshing algorithms based on standard elements (bricks, tetrahedra). Therefore, while the analysis engine can work with any element type, for practical purposes, the current implementation is limited to 7 basic elements: 4 volumetric and 3 surface elements (the segment, although not *flat*, is grouped with surface elements). The 7 elements supported in the current implementation are illustrated in Figure 5.2.

These elements have a unique topology (facets and edges), provided that they are numbered properly. The *surface* elements are not two–dimensional, but three–dimensional entities that can be used to define shells, domes and membranes for example. The *segment* element is special: it can be used to fine–tune the node connectivity matrix by allowing two non–neighboring nodes to share level 1 connectivity.

The base mesh specification for input follows from the two–dimensional implementation, with the main difference being the possibility of defining volumetric and surface elements

Table 5.1: Domain definition (base mesh) input variables for GRAND3.

Variable Name	Type & Size	Description
`NODE`	`array` $N_n \times 3$	Each row p has the nodal coordinates x, y and z for node p.
`ELEM`	`struct`	Structure with fields `V` and/or `S` for *volumetric* and *surface* elements respectively. The numbering scheme is given in Figure 5.2.
`ELEM.V`	`cell` $N_{ve} \times 1$	Entries are row vectors containing the node numbers for each particular *volumetric* element. The total number of volumetric elements is N_{ve}.
`ELEM.S`	`cell` $N_{se} \times 1$	Entries are row vectors containing the node numbers for each particular *surface* element. The total number of surface elements is N_{se}.
`SUPP`	`array` $N_f \times 4$	Each row consists of a node number, fixity x, fixity in y and fixity in z. Any value other than `NaN` specifies fixity. The total number of specified fixities is N_{sup}.
`LOAD`	`array` $N_l \times 4$	Each row consists of a node number, load in x, load in y and load in z. A zero or `NaN` specify no force in that direction.

(Table 5.1). The `ELEM` variable is a structure (or *struct*) with fields `V` and `S` for volumetric and surface elements respectively. The user can pass volumetric, surface or both types of elements in the base mesh. In other words, at least one field (`V` or `S`) must be defined. The total number of volumetric and surface elements are N_{ve} and N_{se} respectively. Therefore, the total number of elements in the base mesh is $N_e = N_{ve} + N_{se}$.

5.2.2 Ground structure generation & collinearity check

The ground structure generation and collinearity check undergo minimal changes compared to the two–dimensional procedure. The creation of the connectivity matrix (and ground structure connectivity level) remain unmodified. The change is in the size of the matrices and vectors involved, where the nodal coordinates are now three–dimensional pairs, and thus

the directional cosines vector for a member becomes (extension of Equation (4.10)):

$$\begin{aligned} \underset{1\times 3}{\mathbf{d}_{p,q}} &= \texttt{NODE}_{q,:} - \texttt{NODE}_{p,:} \\ \underset{1\times 3}{\hat{\mathbf{d}}_{p,q}} &= \frac{\mathbf{d}_{p,q}}{\|\mathbf{d}_{p,q}\|} \end{aligned} \tag{5.3}$$

The collinearity check, just like in the two–dimensional case, is done at each level with the $\mathbf{C}$ matrix (extension of Equation (4.12)):

$$\underset{N_b^{(old)}\times N_b^{(new)}}{\mathbf{C}} = \underset{N_b^{(old)}\times 3}{\mathbf{D}_{old}} \quad \underset{3\times N_b^{(new)}}{\mathbf{D}^{\mathbf{T}}_{new}} , \tag{5.4}$$

where any value in a column higher than $ColTol$ indicates that the member corresponding to that column, should be removed.

5.2.3 Restriction zones

The restriction zones are built from *collision tests*. Collision tests do not aim to find the exact collision point(s), thus differing from *intersection tests*. In general, however, collision tests tend to be computationally cheaper than intersection tests. However, in some cases, in order to determine if there is collision, the exact intersection point is determined.

Literature in this field is rich and varied thanks to the video–game industry, and computer graphics (specifically ray–tracing) literature (Akenine-Möller et al., 2008; Ericson, 2004; Schneider and Eberly, 2002). The intersection of rays and objects is well documented, and the modification from ray to segment is trivial in most cases. The collision primitives currently implemented in GRAND3 are:

1. Box
2. Triangle
3. Quadrangle

4. Sphere

5. Disc

6. Cylinder (infinite length)

7. Rod (finite length cylinder with endcaps)

8. Surface (built from the triangle and quadrangles primitives)

Other primitives can be easily implemented by the user. The collision primitives for the box, triangle, quadrangle and cylinder follow procedures outlined in Ericson (2004) with some modifications. The sphere, disc and rod[2] were developed specifically for GRAND3, although similar procedures are likely to be found in literature given the relatively simple nature of the problem. The collision primitives developed were interactively tested in a variety of scenarios, in order to detect and fix false–positives and false–negatives. Additional details (including source code) on the testing framework are available in Appendix C.

The surface primitive, built from the triangle and quadrangle primitives, is special; a complicated restriction volume can be translated into testing the collision on its surface. This allows the method to address complicated volumes that would be difficult to represent with the already available primitives.

Box primitive

The box primitive is defined with the coordinates of the two extreme vertices; $A_{min} = \{x_{min}\,,\, y_{min}\,,\, z_{min}\}$ and $A_{max} = \{x_{max}\,,\, y_{max}\,,\, z_{max}\}$. Given a segment $\overline{PQ}$, the segment's directional vector is $\mathbf{d} = \overrightarrow{PQ} = Q - P$, and any point X in the segment is defined as $X = P + t\mathbf{d}$, with $0 \leq t \leq 1$.

The segment collides with the box if there is a sub–segment within $\overline{PQ}$ contained inside the box, as shown in Figure 5.3(a). The procedure is better understood in two–dimensions,

[2]Ericson (2004) outlined a procedure for the finite cylinder. However, his derivation is flawed. The book's errata attempts to fix this, but with no success.

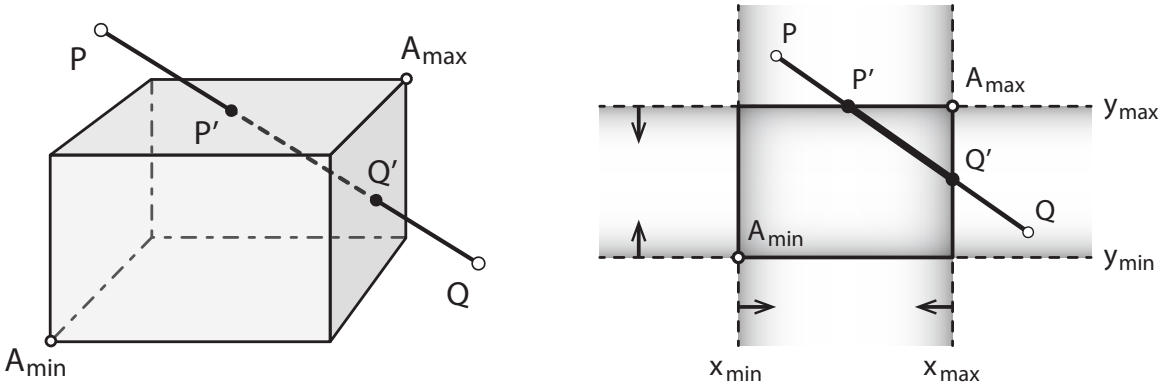


Figure 5.3: Collision test between a box and a segment. (a) Three–dimensional sketch of the box—segment collision test. (b) Two–dimensional simplification of the box—segment collision test (equal to the rectangle—segment test).

where the test becomes the collision test of a rectangle and a segment (Figure 5.3(b)). Once the test in two–dimensions is derived, upscaling to three–dimensional space is trivial.

Defining the sub–segment $\overline{P'Q'}$, with $P' = P + t_{min}\mathbf{d}$ and $Q' = P + t_{max}\mathbf{d}$, the sub–segment is *valid* if $0 \leq t_{min} \leq t_{max} \leq 1$. Initially $t_{min} = 0$ and $t_{max} = 1$, positioning nodes P' and Q' at P and Q respectively. The sub–segment is then clipped by 4 planes (6 in three–dimensional space), corresponding to x_{min}, x_{max}, y_{min} and y_{max} as in Figure 5.3(b). If the sub–segment is still *valid* after the clipping has been done, then the sub–segment is inside the rectangle (box in three–dimensions).

Defining a unit vector in the x direction $\hat{\mathbf{e}}_1 = \{1\,,\,0\,,\,0\}$, the procedure for clipping on the x plane is as follows:

$$t_1 = \frac{A_{min} \cdot \hat{\mathbf{e}}_1 - P \cdot \hat{\mathbf{e}}_1}{\mathbf{d} \cdot \hat{\mathbf{e}}_1} \qquad t_2 = \frac{A_{max} \cdot \hat{\mathbf{e}}_1 - P \cdot \hat{\mathbf{e}}_1}{\mathbf{d} \cdot \hat{\mathbf{e}}_1} \tag{5.5}$$

Depending on the orientation of $\overline{PQ}$, it could occur that $t_1 > t_2$, and in such case their values are switched; $t_1 \leftarrow t_2$ and $t_2 \leftarrow t_1$. Finally, the clipping process is simply:

$$t_{min} \leftarrow \max\left(t_{min}\,,\,t_1\right) \qquad t_{max} \leftarrow \min\left(t_{max}\,,\,t_2\right) \tag{5.6}$$

The process is then repeated for the y plane with $\hat{\mathbf{e}}_2 = \{0\,,\,1\,,\,0\}$, and finally the z plane with $\hat{\mathbf{e}}_3 = \{0\,,\,0\,,\,1\}$. The segment collides with the box if $t_{min} \leq t_{max}$ after all the clipping has been carried out. Incidentally, this procedure can address the accidental case where A_{min} and A_{max} are reversed.

Triangle primitive

Given a segment $\overline{PQ}$, intersecting the plane defined by points A, B and C in space at a point W (Figure 5.4). The segment intersects the triangle $\triangle ABC$ if point W lies inside the triangle.

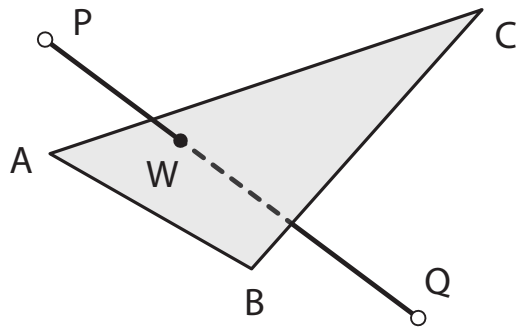


Figure 5.4: Collision test between a triangle and a segment.

One possible solution is to find the point W, and then check if such point is inside the triangle. Point W is found to be inside the triangle if it is a convex combination of points A, B and C: $W = \alpha_1 A + \alpha_2 B + \alpha_3 C$, with $\alpha_1 + \alpha_2 + \alpha_3 = 1$. An alternative procedure considers the triangle to be arranged counterclockwise: point W is inside the triangle if it is located to the left of all of the triangle's edges. Extending to any triangle arrangement (clockwise or counterclockwise): point W is inside the triangle if it is located to the same side for all of the triangle's edges. Based on this idea, the volumes of the three distinct tetrahedra can be defined by segment $\overline{PQ}$, and each one of the triangle's edges. These volumes can be computed by triple products, with the sign of these triple products depending on the manifold orientation of the three vectors defining the tetrahedron (in volume calculations an

absolute value is often used). If point W is found to the same side of all edges, then the sign of these volume calculations (triple products) should be the same.

Defining the segment's directional vector as $\mathbf{d} = \overrightarrow{PQ} = Q - P$, then any point X within the segment is defined as $X = P + t\mathbf{d}$, with $0 \leq t \leq 1$. The normal to the plane of the triangle is $\mathbf{n} = \overrightarrow{AB} \times \overrightarrow{AC}$. The intersection point with the plane of the triangle W is defined as $W = P + t_w\mathbf{d}$, with:

$$t_w = \frac{\overrightarrow{PA} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \,. \tag{5.7}$$

The triple products are defined as:

$$v_1 = \left[\mathbf{d}\ \overrightarrow{PC}\ \overrightarrow{PB}\right] \qquad v_2 = \left[\mathbf{d}\ \overrightarrow{PA}\ \overrightarrow{PC}\right] \qquad v_3 = \left[\mathbf{d}\ \overrightarrow{PB}\ \overrightarrow{PA}\right] \tag{5.8}$$

Taking $\mathbf{e}_1 = \mathbf{d} \times \overrightarrow{PC}$ and $\mathbf{e}_2 = \mathbf{d} \times \overrightarrow{PB}$, then the triple products become:

$$v_1 = \quad \mathbf{e}_1 \cdot \overrightarrow{PB} \qquad v_2 = -\mathbf{e}_1 \cdot \overrightarrow{PA} \qquad v_3 = \quad \mathbf{e}_2 \cdot \overrightarrow{PA} \tag{5.9}$$

Finally, the segment $\overline{PQ}$ intersects the triangle $\triangle ABC$ if and only if:

$$0 \leq t_w \leq 1 \qquad \text{and} \qquad \text{sign}\,(v_1) = \text{sign}\,(v_2) = \text{sign}\,(v_3) \tag{5.10}$$

Quadrangle primitive

Splitting the quadrangle into two triangles, the quadrangle is really an extension of the triangle primitive case. It is assumed that the quadrangle is flat and all 4 points lie (approximately) in the same plane. Compared to two complete triangle tests, there is a potential computational saving if one of the two triangles is chosen early in the calculations (triangles $\triangle AB_1C$ and $\triangle ACB_2$ in Figure 5.5).

Defining the segment's directional vector as $\mathbf{d} = \overrightarrow{PQ} = Q - P$, then any point X within the segment is defined as $X = P + t\mathbf{d}$, with $0 \leq t \leq 1$. The normal to the plane of the

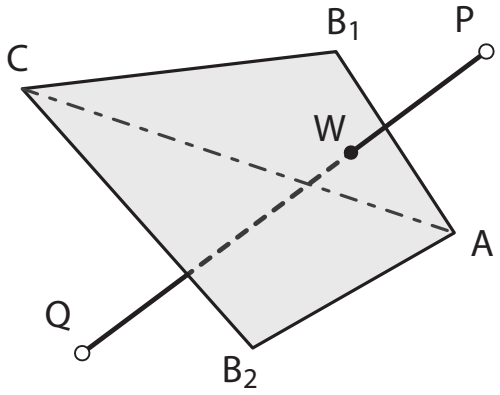


Figure 5.5: Collision test between a quadrangle and a segment.

quadrangle is $\mathbf{n} = \overrightarrow{AB_1} \times \overrightarrow{AB_2}$. The intersection point with the plane of the quadrangle is defined as $W = P + t_w\mathbf{d}$, with:

$$t_w = \frac{\overrightarrow{PA} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \,. \tag{5.11}$$

The triple products are defined as:

$$\begin{aligned} v_a = & \quad \left[\mathbf{d}\ \overrightarrow{PC}\ \overrightarrow{PB_1}\right] &= & \quad \mathbf{e}_1 \cdot \overrightarrow{PB_1} \\ v_b = & \quad \left[\mathbf{d}\ \overrightarrow{PA}\ \overrightarrow{PB_2}\right] &= & \quad \mathbf{e}_1 \cdot \overrightarrow{PB_2} \\ v_2 = & -\left[\mathbf{d}\ \overrightarrow{PB}\ \overrightarrow{PA}\right] &= & -\mathbf{e}_1 \cdot \overrightarrow{PA}\,, \end{aligned} \tag{5.12}$$

with $\mathbf{e}_1 = \mathbf{d} \times \overrightarrow{PC}$. If the segment collides with a triangle, then all triple products must have the same sign. Thus, the correct triangle can be chosen at this stage, and the third (and last) triple product can be computed:

$$\mathbf{e}_2 \ = \ \begin{cases} \mathbf{d} \times \overrightarrow{PB_1} & \text{if} \quad \text{sign}\,(v_a) = \text{sign}\,(v_2) \\ \mathbf{d} \times \overrightarrow{PB_2} & \text{if} \quad \text{sign}\,(v_b) = \text{sign}\,(v_2) \end{cases} , \tag{5.13}$$

with the third triple product being:

$$v_3 = \mathbf{e}_2 \cdot \overrightarrow{PA} \tag{5.14}$$

The segment collides with the quadrangle $\square AB_1CB_2$ if and only if:

$$0 \leq t_w \leq 1 \qquad \text{and} \qquad \text{sign}\,(v_2) = \text{sign}\,(v_3) \tag{5.15}$$

Sphere primitive

Defining the segment's directional vector as $\mathbf{d} = \overrightarrow{PQ} = Q - P$, then any point X in the segment is defined as $X = P + t\mathbf{d}$, with $0 \leq t \leq 1$. The segment intersects the sphere if any of the following three criteria is met (Figure 5.6):

1. Point P is inside the sphere.
2. Point Q is inside the sphere.
3. The point in the segment that is closest to the sphere's center is between P and Q and inside the sphere.

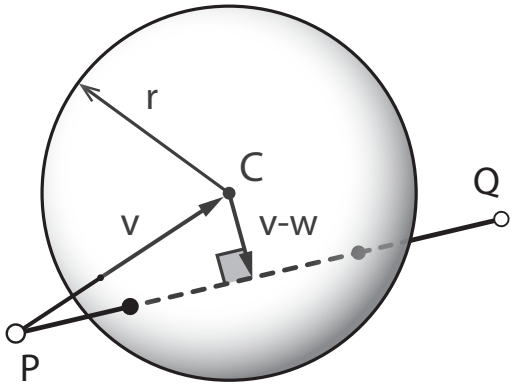


Figure 5.6: Collision test between a sphere and a segment.

Defining a vector $\mathbf{v} = C - P$, then there is collision according to the first criteria if:

$$\mathbf{v} \cdot \mathbf{v} \leq r^2 \tag{5.16}$$

Similarly, there is collision according to the second criteria if:

$$(\mathbf{v}-\mathbf{d})\cdot(\mathbf{v}-\mathbf{d}) = \mathbf{v}\cdot\mathbf{v} - 2\mathbf{v}\cdot\mathbf{d} + \mathbf{d}\cdot\mathbf{d} \leq r^2 \tag{5.17}$$

The closest point in the line defined by P and Q to the sphere's center, measured from point P is:

$$\mathbf{w} = \frac{\mathbf{v}\cdot\mathbf{d}}{\mathbf{d}\cdot\mathbf{d}}\mathbf{d} \tag{5.18}$$

This point is inside the sphere if:

$$(\mathbf{v}-\mathbf{w})\cdot(\mathbf{v}-\mathbf{w}) = \left(\mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{d}}{\mathbf{d}\cdot\mathbf{d}}\mathbf{d}\right)\cdot\left(\mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{d}}{\mathbf{d}\cdot\mathbf{d}}\mathbf{d}\right) \leq r^2\,, \tag{5.19}$$

with this point inside the segment $\overline{PQ}$ if and only if $0 \leq \mathbf{w}\cdot\mathbf{d} \leq \mathbf{d}\cdot\mathbf{d}$, which is an equivalent expression to $0 \leq t \leq 1$.

Disc primitive

Defining the segment's directional vector as $\mathbf{d} = \overrightarrow{PQ} = Q - P$, then any point X in the segment is defined as $X = P + t\mathbf{d}$, with $0 \leq t \leq 1$.

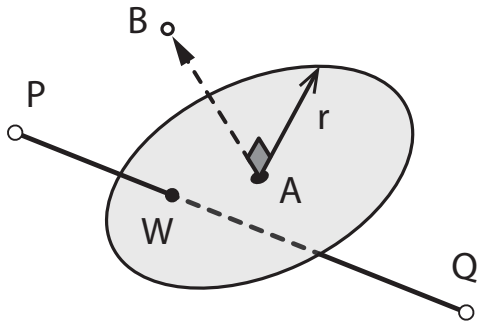


Figure 5.7: Collision test between a disc and a segment.

The disc is centered at point A, and the normal points towards a point B as in Figure 5.7. The normal to the plane of the disc is $\mathbf{n} = B - A$. Defining $\mathbf{v} = \overrightarrow{AP} = P - A$, the

intersection with the disc's plane is found at a point $W = P + t_w\mathbf{d}$:

$$t_w = -\frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{d}\cdot\mathbf{n}} \tag{5.20}$$

The segment collides with the disc if the distance between point W and the disc's center A is less than or equal to the disc's radius:

$$\overrightarrow{AW} = W - A = P - A + t_w\mathbf{d} = \mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{d}\cdot\mathbf{n}}\mathbf{d} \tag{5.21}$$

Finally, the segment collides with the disc if and only if:

$$\overrightarrow{AW}\cdot\overrightarrow{AW} = \left(\mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{d}\cdot\mathbf{n}}\right)\cdot\left(\mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{d}\cdot\mathbf{n}}\right) \leq r^2\,, \tag{5.22}$$

with $0 \leq t_w \leq 1$.

Cylinder primitive (infinite cylinder)

Defining the segment's directional vector as $\mathbf{d} = \overrightarrow{PQ} = Q - P$, then any point X in the segment is defined as $X = P + t\mathbf{d}$, with $0 \leq t \leq 1$. The segment is found to collide with the (infinite) cylinder if:

1. The segment collides with the cylinder's surface.

2. The segment is completely contained within the cylinder.

The cylinder's axis is defined by $\mathbf{n} = B - A$ as in Figure 5.8. The intersection with the cylinder's surface is found at points $W = P + t_w\mathbf{d}$. Defining the radial vector of length r from the cylinder's axis to point W as $\mathbf{m}$, then:

$$0 = \mathbf{m}\cdot\mathbf{m} - r^2 = \left(\overrightarrow{AW} - \frac{\overrightarrow{AW}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\mathbf{n}\right)\cdot\left(\overrightarrow{AW} - \frac{\overrightarrow{AW}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\mathbf{n}\right) - r^2 \tag{5.23}$$

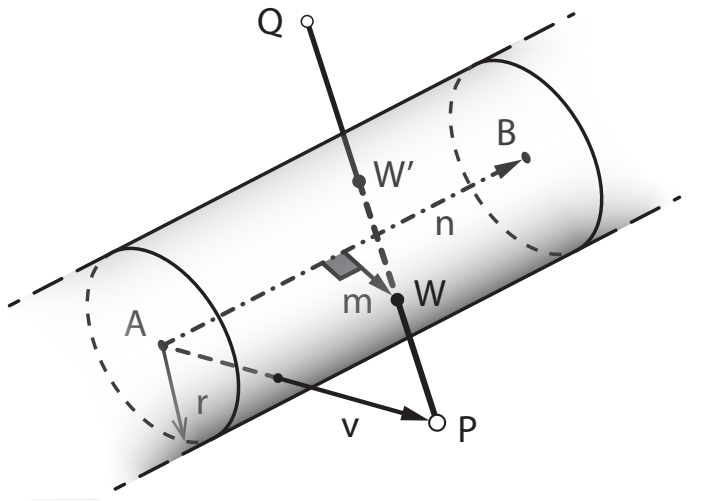


Figure 5.8: Collision test between a cylinder (infinite length) and a segment.

Defining $\mathbf{v} = \overrightarrow{AP} = P - A$, and expanding $\overrightarrow{AW}$ in terms of $\mathbf{v}$, $\mathbf{d}$ and t_w:

$$0 = \left((\mathbf{v} + t_w \mathbf{d}) - \left[\frac{(\mathbf{v} + t_w \mathbf{d}) \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}} \right] \mathbf{n} \right) \cdot \left((\mathbf{v} + t_w \mathbf{d}) - \left[\frac{(\mathbf{v} + t_w \mathbf{d}) \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}} \right] \mathbf{n} \right) - r^2 \,, \tag{5.24}$$

where terms can be ordered to obtain a quadratic equation for t_w:

$$0 = \left[\mathbf{d} \cdot \mathbf{d} - \frac{(\mathbf{d} \cdot \mathbf{n})^2}{\mathbf{n} \cdot \mathbf{n}} \right] t_w^2 + 2 \left[\mathbf{v} \cdot \mathbf{d} - \frac{(\mathbf{d} \cdot \mathbf{n})(\mathbf{v} \cdot \mathbf{n})}{\mathbf{n} \cdot \mathbf{n}} \right] t_w + (\mathbf{v} \cdot \mathbf{v}) - \frac{(\mathbf{v} \cdot \mathbf{n})^2}{\mathbf{n} \cdot \mathbf{n}} - r^2 \tag{5.25}$$

Multiplying by $\mathbf{n} \cdot \mathbf{n}$, the quadratic equation becomes:

$$\begin{aligned} 0 &= a t_w^2 + b t_w + c \\ a &= (\mathbf{n} \cdot \mathbf{n})(\mathbf{d} \cdot \mathbf{d}) - (\mathbf{d} \cdot \mathbf{n})^2 \\ \frac{b}{2} &= (\mathbf{n} \cdot \mathbf{n})(\mathbf{v} \cdot \mathbf{d}) - (\mathbf{d} \cdot \mathbf{n})(\mathbf{v} \cdot \mathbf{n}) \\ c &= (\mathbf{n} \cdot \mathbf{n}) \left[(\mathbf{v} \cdot \mathbf{v}) - r^2 \right] - (\mathbf{v} \cdot \mathbf{n})^2 \,, \end{aligned} \tag{5.26}$$

with solutions given by:

$$t_w = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-\frac{b}{2} \pm \sqrt{\left(\frac{b}{2}\right)^2 - ac}}{a} \tag{5.27}$$

The (infinite) line defined by the segment does not intersect the cylinder if the discriminant in Equation (5.27) is negative (i.e. $(b/2)^2 - ac < 0$). If the discriminant is positive, then an additional check must be made to ensure the intersection point is within the segment $\overline{PQ}$: The segment collides with the cylinder if $0 \leq t_w \leq 1$ for any of the two roots from Equation 5.27, corresponding to points W and W' in Figure 5.8.

Finally, the segment is completely contained inside the cylinder, if the distance from the cylinder's axis to point P is less than or equal to the radius r:

$$\left(\mathbf{v} - \frac{\mathbf{v} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}}\mathbf{n}\right) \cdot \left(\mathbf{v} - \frac{\mathbf{v} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}}\mathbf{n}\right) \leq r^2 \tag{5.28}$$

Rod primitive

The rod primitive is a combination of the infinite cylinder and disc primitives with some minor modifications. The segment collides with the rod if any of the following 4 situations occur:

- The segment collides with the finite cylinder's surface.
- The segment collides with the A endcap (disc).
- The segment collides with the B endcap (disc).
- The segment is fully contained within the rod.

The collision with the finite cylinder's surface begins from the test primitive for the infinite cylinder outlined in Equations (5.26) and (5.27). In addition, the intersection points W and W' (Figure 5.9) must be in the surface between the endcaps. Thus, an additional

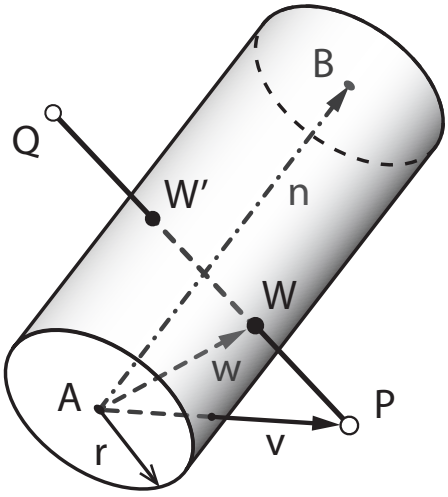


Figure 5.9: Collision test between a rod (finite cylinder with endcaps) and a segment.

check is required; the segment collides with the finite cylinder's surface if:

$$\begin{aligned} 0 \leq & \mathbf{w} \cdot \mathbf{n} \leq \mathbf{n} \cdot \mathbf{n} \\ 0 \leq & (\mathbf{v} + t_w \mathbf{d}) \cdot \mathbf{n} \leq \mathbf{n} \cdot \mathbf{n} \end{aligned} \tag{5.29}$$

for any of the two roots of t_w from Equation (5.27), with $0 \leq t_w \leq 1$.

The collision with the endcaps A and B follow the procedure for the disc primitive. Equation (5.22) can be used with no modification to test the collision against endcap A. The B endcap is analogous to the endcap A; the segment collides with endcap B if:

$$\overrightarrow{BW} \cdot \overrightarrow{BW} = \left(\mathbf{v} + \frac{\mathbf{n} \cdot \mathbf{n} - \mathbf{v} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \right) \cdot \left(\mathbf{v} + \frac{\mathbf{n} \cdot \mathbf{n} - \mathbf{v} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \right) \leq r^2 \,, \tag{5.30}$$

with:

$$0 \leq \left(t_w = \frac{\mathbf{n} \cdot \mathbf{n} - \mathbf{v} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \right) \leq 1 \tag{5.31}$$

Finally, if the segment is completely contained in the rod, then point P must be inside the rod. In other words, if the distance from point P to the cylinder's axis is less than or

equal to r:

$$\left(\mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\mathbf{n}\right)\cdot\left(\mathbf{v} - \frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\mathbf{n}\right) \leq r^2\,, \tag{5.32}$$

with an additional check to verify point A is between the endcaps:

$$0 \leq \frac{\mathbf{v}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}} \leq 1 \tag{5.33}$$

Surface primitive

The surface primitive builds from the base of the triangle and quadrangle primitives. The surface primitive can handle any surface provided that it is tessellated (discretized) and the points in each facet lie (approximately) in the same plane. In addition, it is assumed that all facets are convex in their own plane. An example of a tessellated surface is shown in Figure 5.10: the surface was tessellated using triangles and quadrangles. The inputs for this collision primitive are:

- A matrix of nodes `RNODE` of size $N_{rn} \times 3$, where N_{rn} is the number of nodes in the collision surface.
- A list (cell) with facet connectivity `RFACE` of size $N_{rf} \times 1$, where N_{rf} is the number of facets in the collision surface. Each entry in `RFACE` is a row vector with nodal connectivity (based on `RNODE`).

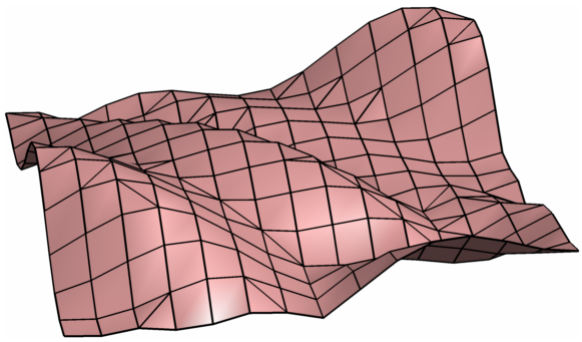


Figure 5.10: Collision surface example: Surface is tessellated into triangles and quadrangles.

The surface collision primitive can address facets with more than 4 nodes (flat polygons), provided that all the nodes lie in (approximately) the same plane. This polygon will be subdivided into triangles and evaluated sequentially.

5.2.4 Plotting scheme

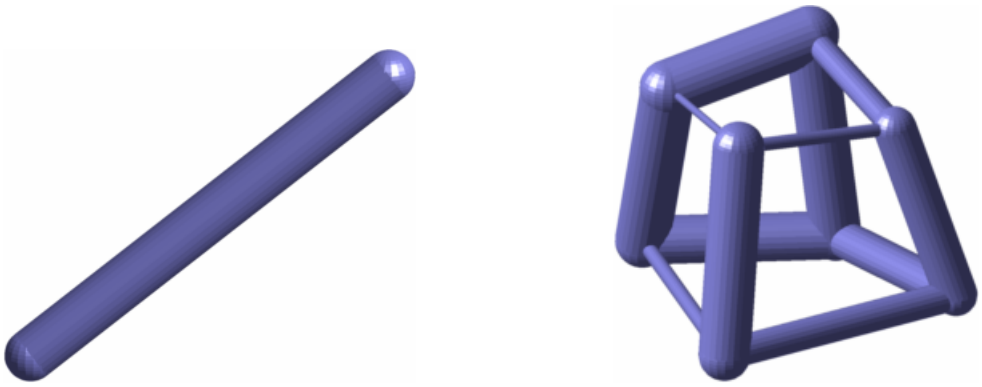
The analysis in the current implementation (Equation (5.2) does not consider the shape of member's section. Issues such as the cost of connections, local buckling and manufacturing costs are not taken into account. Thus, given a known cross–sectional area and no information on the section type, the section is assumed to be circular (cylinders).

The optimal cross–sectional areas are obtained for $\sigma_T = 1$, and the solution must be scaled for values other than unity. Therefore, the members in the optimal structure are represented as cylinders with radii equal to $r_i = c\sqrt{a_i}$, where c is a scaling constant. Analogous to the two–dimensional implementation, only the members with cross–sectional areas $a_i > (\mathit{Cutoff})\,(a_{max})$ are plotted: this may cause a truss cord to abruptly end mid–air in the resulting figure, which is not accurate, but is merely a plotting artifact.

Nodes with one or more members above the cutoff cross–sectional area will be represented graphically by a sphere. The sphere's radius is equal to the largest radius of all members connected to that node. Therefore, a single member connecting two nodes is represented as in Figure 5.11(a). The case of multiple members with varying cross–sectional areas is shown in Figure 5.11(b).

5.3 Verification using known analytical solutions

The following examples aim to verify GRAND3 by approximating optimal closed–form solutions. Unless otherwise stated, the stress limit ratio is $\kappa = 1.0$ with $\sigma_T = 1$, the collinear tolerance is $\mathit{ColTol} = 0.999999$, and the plotting cutoff is $\mathit{Cutoff} = 0.005$.



(a) (b)

Figure 5.11: Plotting scheme sample for three–dimensional ground structures. (a) Single member connecting two nodes. (b) Multiple members with varied cross–sectional areas.

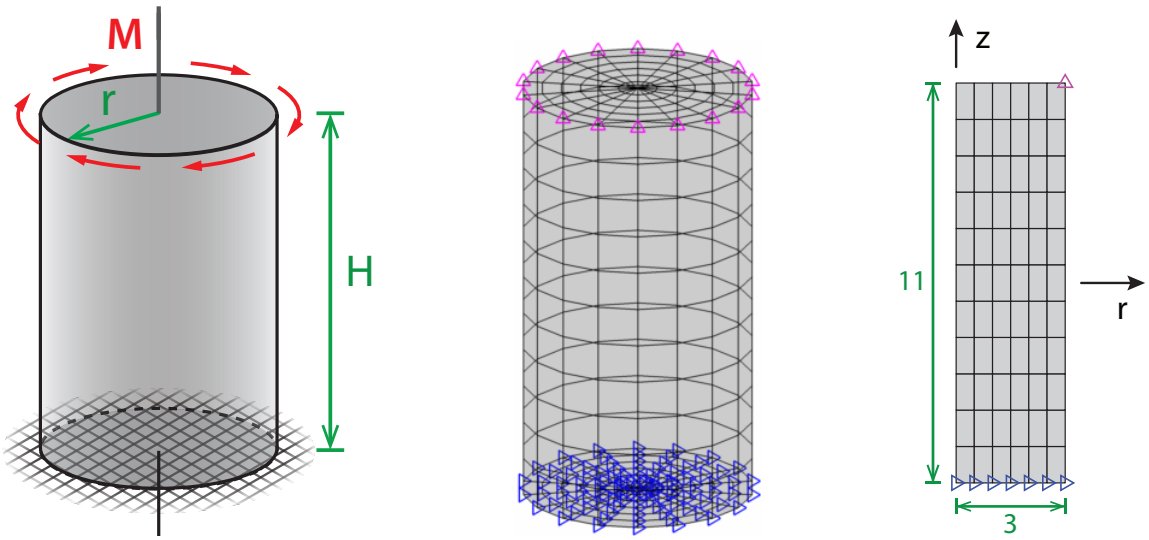
5.3.1 Torsion cylinder

A cylindrical domain of radius r and height H is subjected to an end moment distributed over the outer ring of the end cap (Figure 5.12(a)). The domain is fully supported on the bottom end cap. This problem is of interest since the analytical volume for the theoretical optimal structure can be derived. A moment pair causes a pure–shear condition on the cylinder surface. Therefore, the principal stress lines are oriented at $\pm\pi/4$. The members in the optimal solution follow the lines of principal stresses (Michell, 1904; Hencky, 1923; Hemp, 1973). A single fiber following a principal stress line has a length $l_f = \sqrt{2}H$. The force in the fiber due to the moment pair is $n_f = \sqrt{2}M/4\pi r^2$ for tension, and $-n_f$ for compression. Finally, the volume of the optimal cylinder in torsion is:

$$
\begin{aligned}
V_{opt} &= \left[\frac{(n_f)\,(l_f)}{\sigma_T} - \frac{(-n_f)\,(l_f)}{\sigma_C}\right](2\pi r) \\
V_{opt} &= \left(\frac{\sqrt{2}M}{4\pi r^2}\right)\left(\sqrt{2}H\right)\left[\frac{1}{\sigma_T} + \frac{1}{\sigma_C}\right](2\pi r) \\
V_{opt} &= \frac{MH}{r}\left[\frac{1}{\sigma_T} + \frac{1}{\sigma_C}\right]
\end{aligned}
\tag{5.34}
$$

Using cylindrical coordinates, the model is discretized in $N_z \times N_r \times N_\theta$ elements cor-

responding to the z, r and θ coordinate axes respectively. The problem is analyzed for the specific case where $H = 11$, $r = 3$ and $M = 5$. A sample mesh discretized with $\{N_z, N_r, N_\theta\} = \{12, 6, 16\}$ is shown in Figure 5.12(b). An axisymmetrical plot of this mesh is given in Figure 5.12(c). The convergence of the ground structure method with refinement of the base mesh is given in Table 5.2 and Figure 5.13(a). There is a convergence towards the optimal volume $V_{opt} = 36.6667$. As an example, one of these solutions is plotted in Figure 5.13(b).



(a) (b) (c)

Figure 5.12: Torsion cylinder problem. (a) Domain definition, loading and supports. (b) Sample mesh for $H = 11$, $r = 3$ discretized with $N_z = 11$, $N_r = 6$ and $N_\theta = 18$. (c) Axisymmetric plot of the sample mesh with $H = 11$, $r = 3$ and $N_z = 11$, $N_r = 6$.

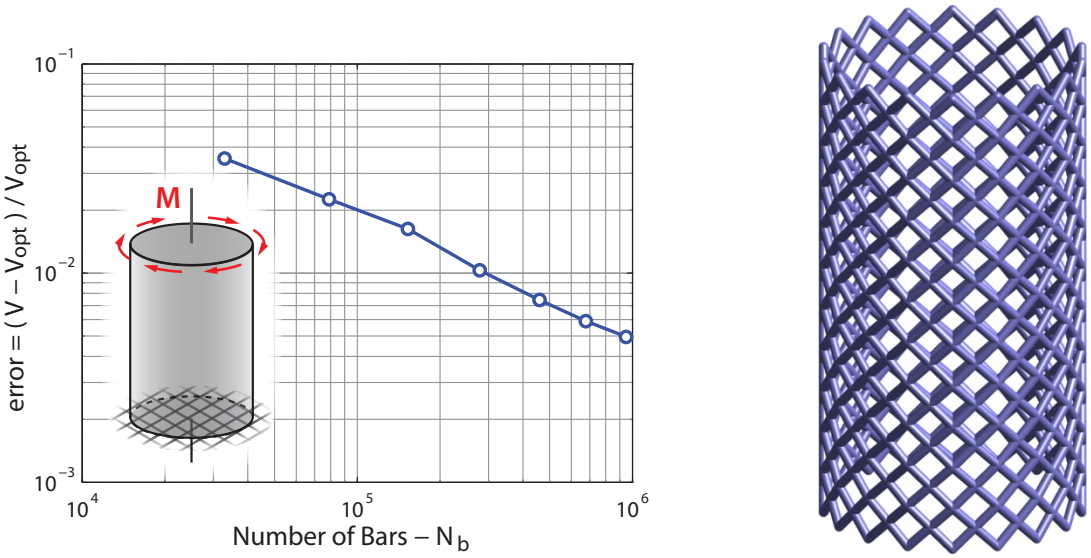
5.3.2 Torsion cone

A capped cone domain of height $H = 10$ with lower and upper radius $r_L = 7$ and $r_U = 2$ respectively, is subjected to an end moment $M = 3$ distributed over the outer ring at the upper end cap (Figure 5.14(a)). The analytical solution for this problem was derived by

Table 5.2: Convergence for a cylinder under torsion with $M = 5$, $H = 11$ and $r = 3$. Ground structures are generated with $Lvl = 3$. The optimal volume is $V_{opt} = 36.6667$.

N_z	N_r	N_θ	N_e	N_n	N_b	Volume V	LP_{iter}	Runtime* [min]
7	4	12	336	392	32,911	37.9628	8	0.049
9	5	15	675	760	78,954	37.4937	9	0.283
11	6	18	1,188	1,308	152,795	37.2637	11	0.803
13	7	22	2,002	2,170	278,467	37.0453	94	16.06
15	8	26	3,120	3,344	458,811	36.9395	11	10.13
17	9	29	4,437	4,716	677,370	36.8830	10	25.21
19	10	32	6,080	6,420	950,419	36.8486	11	52.35

* Runtimes measured on an Intel Xeon E3-1245 with 32GB of RAM with MATLAB R2013b.



(a) (b)

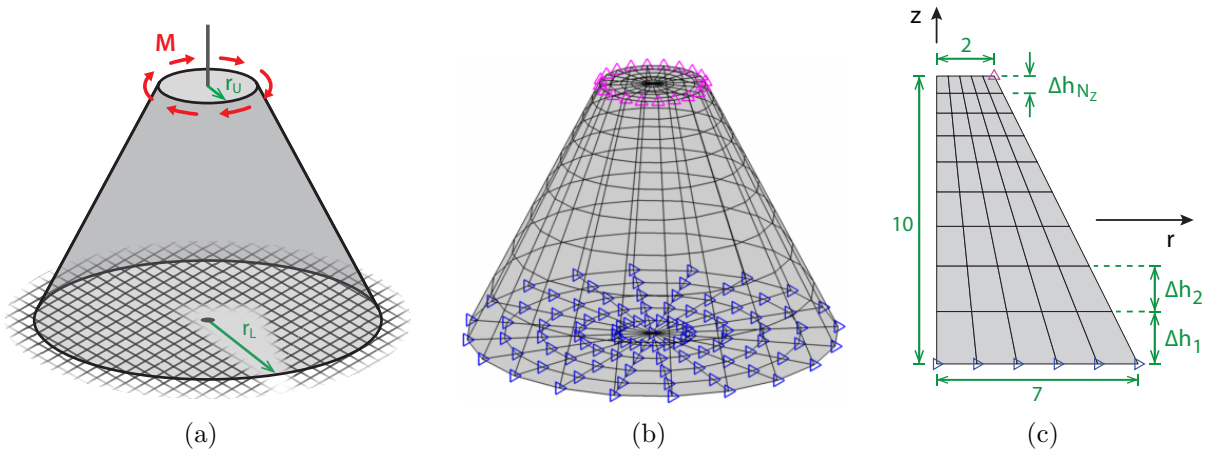
Figure 5.13: Cylinder domain under torsion. (a) Convergence with base mesh refinement. (b) Solution obtained for $N_b = 152,795$, generated from a cylindrical domain with $N_e = 1,188$, $N_n = 1,308$ and $Lvl = 3$.

Lewiński (2004), by constraining the solution to exist in the cone's surface:

$$V_{opt} = M \frac{\sqrt{H^2 + (r_L - r_U)^2}}{r_L - r_U} \log\left(\frac{r_L}{r_U}\right) \left[\frac{1}{\sigma_T} + \frac{1}{\sigma_C}\right] = 16.8076 \tag{5.35}$$

Using cylindrical coordinates, the model is discretized in $N_z \times N_r \times N_\theta$ elements corresponding to the z, r and θ coordinate axes respectively. An axisymmetrical plot of this mesh

is given in Figure 5.14(c). In an effort to preserve the aspect ratio of the elements in the mesh, the spacing in the z direction is such that $\Delta h_{i+1}/\Delta h_i = \lambda$, with λ being constant. A sample mesh discretized with $\{N_z, N_r, N_\theta\} = \{9, 5, 20\}$ and $\lambda = 0.87006$ is shown in Figure 5.14(b). The convergence of the ground structure method with refinement of the base mesh is given in Table 5.3 and Figure 5.15(a). As an example, one of these solutions is plotted in Figure 5.15(b). The increasingly refined solutions converge smoothly toward the optimum, with a relatively small oscillation appearing when the number of bars is $N_b > 400,000$. The reason behind this oscillation being that the node positions do not precisely match the locations dictated by the analytical closed–form solution. In other words, this is caused by the aspect ratio of the discretization, influenced by N_z, N_θ and λ. However, the overall trend does converge to the analytical optimum as expected.



(a) (b) (c)

Figure 5.14: Torsion cone problem. (a) Domain definition, loading and supports. (b) Sample mesh for $H = 10$, $r_L = 7$ and $r_U = 2$ discretized with $N_z = 9$, $N_r = 5$, $N_\theta = 20$ and $\lambda = 0.870058$. (c) Axisymmetric plot of the sample mesh with $H = 10$, $r_L = 7$, $r_U = 2$, $N_z = 9$, $N_r = 5$ and $\lambda = 0.870058$.

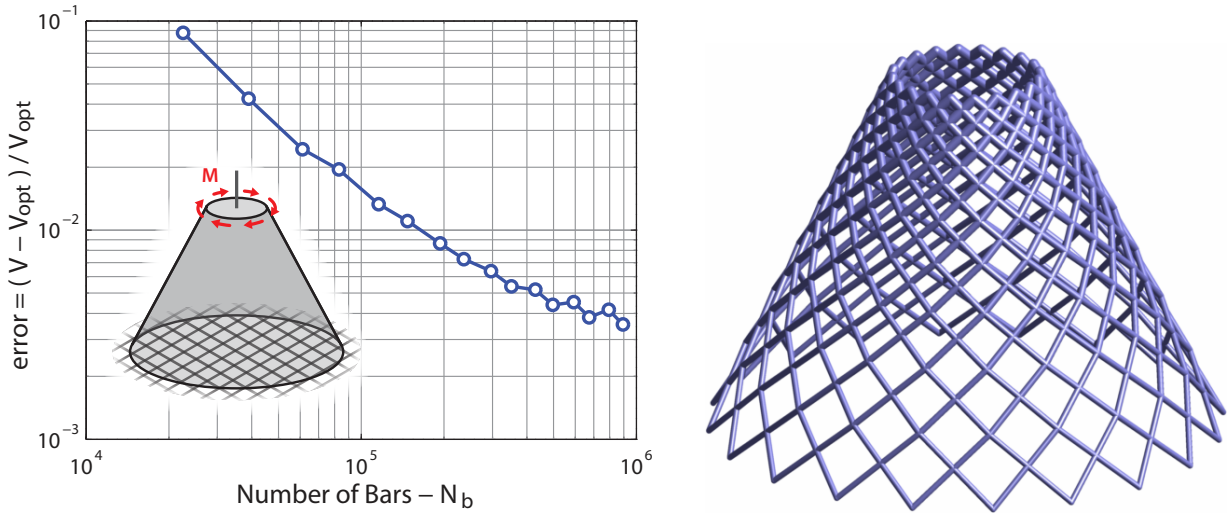
5.3.3 Torsion sphere (orthogonal domain)

The optimal structure for transmitting a torsional moment is a *ball*, provided that the domain is large enough to allow the full solution to develop (Figure 5.1(a)). In contrast, the

Table 5.3: Convergence for a capped cone under torsion with $M = 3$, $H = 10$, $r_L = 7$ and $r_U = 2$. Ground structures are generated with $N_r = 5$ and $Lvl = 3$. The optimal volume is $V_{opt} = 16.8076$.

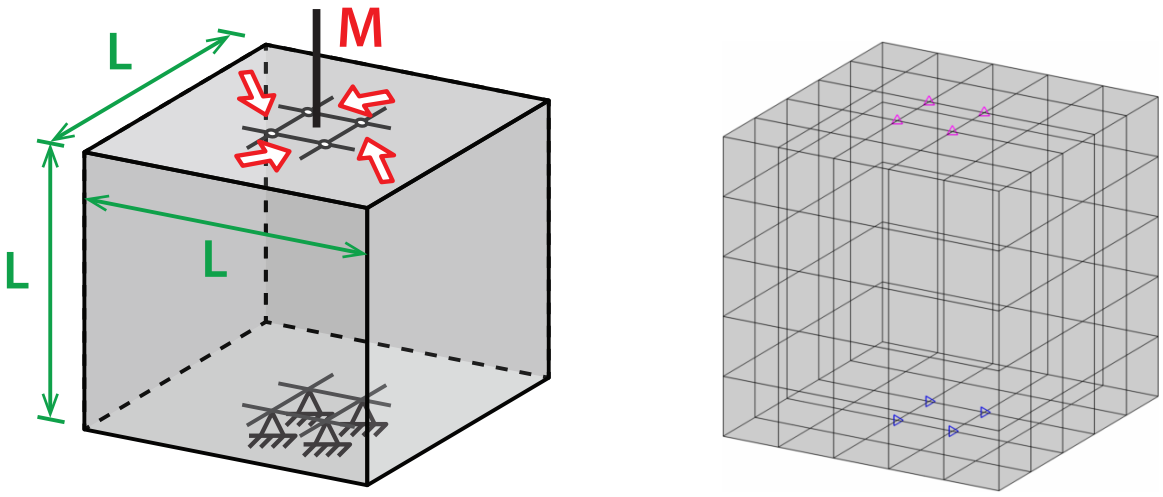
N_z	N_θ	λ	N_e	N_n	N_b	Volume V	LP_{iter}	Runtime* [min]
5	9	0.778371	225	276	22,532	18.2822	10	0.027
6	12	0.811563	360	427	38,946	17.5218	11	0.071
7	15	0.836134	525	608	61,072	17.2178	10	0.171
8	17	0.855050	680	774	82,849	17.1354	11	0.332
9	20	0.870058	900	1,010	115,789	17.0310	11	0.609
10	22	0.882253	1,100	1,221	147,058	16.9935	153	8.510
11	25	0.892358	1,375	1,512	193,686	16.9526	53	6.492
12	27	0.900868	1,620	1,768	235,938	16.9300	10	2.115
13	30	0.908131	1,950	2,114	296,383	16.9146	10	3.212
14	32	0.914404	2,240	2,415	350,830	16.8982	10	3.970
15	35	0.919875	2,625	2,816	429,220	16.8947	12	5.873
16	37	0.924689	2,960	3,162	497,407	16.8813	12	7.063
17	40	0.928958	3,400	3,618	592,377	16.8837	14	11.01
18	42	0.932769	3,780	4,009	675,462	16.8720	18	17.14
19	45	0.936192	4,275	4,520	792,874	16.8776	113	121.7
20	47	0.939283	4,700	4,956	892,456	16.8670	116	162.5

* Runtimes measured on an Intel Xeon E3-1245 with 32GB of RAM with MATLAB R2013b.



(a) (b)

Figure 5.15: Capped cone domain under torsion. (a) Convergence with base mesh refinement. (b) Solution obtained for $N_b = 115,789$, generated using a cylindrical–coordinate domain with $N_e = 900$, $N_n = 1,010$ and $Lvl = 3$.



(a) (b)

Figure 5.16: Torsion sphere problem modeled using an orthogonal domain. (a) Domain definition, loading and supports. (b) Sample mesh with $N = 5$.

example in Section 5.3.1 (torsion cylinder) addresses a similar problem, but with a constrained domain. With no previous knowledge of the optimal solution, it is reasonable to use a regular and orthogonal base mesh. The moment (and support) is applied at the four nodes closest to the poles, as shown in Figures 5.16(a) and 5.16(b).

The domain is discretized using $N \times N \times N$ elements. Considering a domain with $L = 1$ and $M = 1$, sample solutions for two different discretizations are shown in Figures 5.18(a) and 5.18(b). The convergence of the ground structure method for increasingly refined base meshes is given in Table 5.4 and Figure 5.17. It should be noted that distance from the loaded and supported nodes to the pole axis decreases with refinement, i.e. the angle ϕ_F increases with a decrease of the element size in the base mesh (Figure 5.20(c)). Thus, the optimal volume V_{opt} changes with the discretization. For a regular structured–orthogonal cube mesh of dimension L, discretized with N elements in each direction (Figure 5.16), the

angle ϕ_F is:

$$
\begin{aligned}
dx &= \frac{L}{N} \\
r_F &= \frac{1}{2}\sqrt{dx^2 + dx^2} = \frac{\sqrt{2}L}{2N} \\
\phi_F &= \operatorname{atan}\left(\frac{L/2}{r_F}\right) = \operatorname{atan}\left(\frac{\sqrt{2}N}{2}\right) \\
R &= \sqrt{\left(\frac{L}{2}\right)^2 + r_F^2} = \frac{L}{2N}\sqrt{N^2+2}\,,
\end{aligned}
\tag{5.36}
$$

where dx is the dimension of a single hexahedral element in the base mesh, and r_F is the distance from the pole to the loaded nodes (radius of the moment application circle).

The poor convergence rate observed with mesh refinement, and even worse with connectivity level, is attributed to a number of reasons:

- This is a particularly unfavorable scenario for the method: approximating a sphere with a *box*.

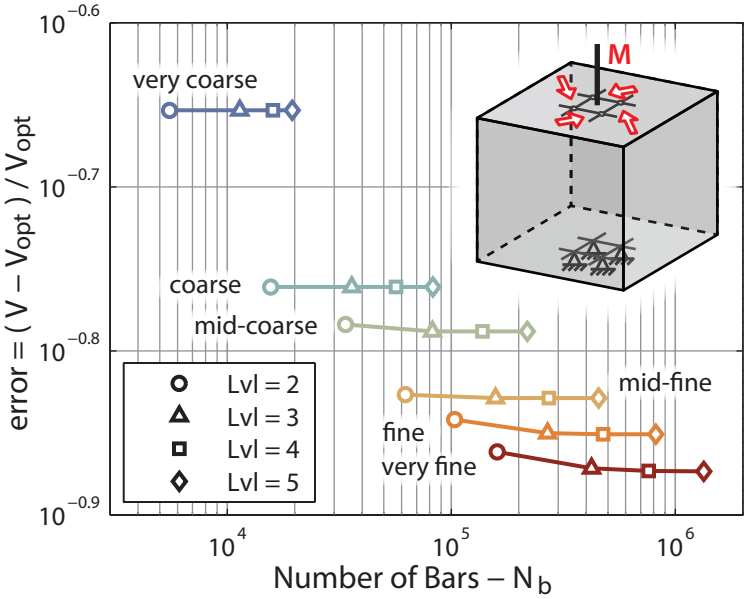
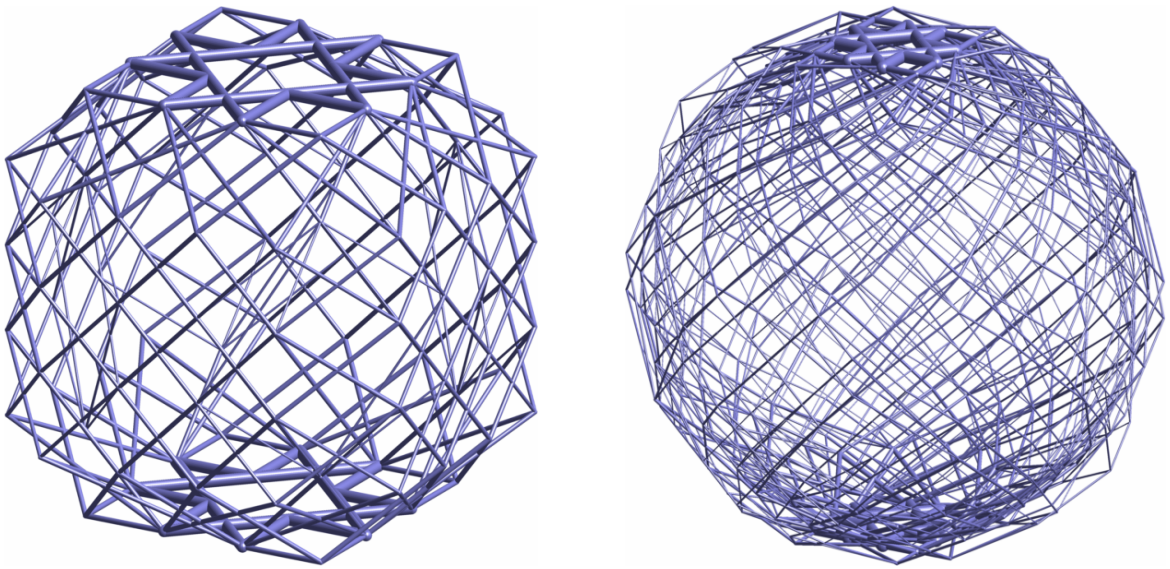


Figure 5.17: Convergence to the optimal solution of increasingly refined regular orthogonal base meshes under torsion. Increasing the ground structure connectivity level does not improve the quality of the solution in this case.

Table 5.4: Convergence for a regular orthogonal domain of side $L = 1$ under torsion with $M = 1$, for different meshes with varied connectivity levels.

Base mesh	N	N_e	N_n	R	ϕ_F	Lvl	N_b	V_{opt}	Volume V	LP_{iter}	Runtime* [min]
very coarse	5	125	216	0.5196	1.2952	2	5,540	7.9017	9.6571	10	0.006
						3	11,372		9.6571	11	0.016
						4	15,980		9.6571	11	0.024
						5	19,508		9.6571	11	0.025
coarse	7	343	512	0.5101	1.3714	2	15,652	9.2101	10.8069	12	0.038
						3	35,932		10.8069	13	0.107
						4	56,668		10.8069	13	0.203
						5	82,804		10.8069	13	0.255
mid-coarse	9	729	1,000	0.5061	1.4149	2	33,804	10.1997	11.8774	14	0.397
						3	82,356		11.8615	17	0.583
						4	137,652		11.8615	16	1.156
						5	218,652		11.8615	17	2.358
mid-fine	11	1,331	1,728	0.5041	1.4429	2	62,348	10.9943	12.6330	16	1.585
						3	157,604		12.6252	17	2.561
						4	272,804		12.6252	17	3.658
						5	454,748		12.6252	18	9.408
fine	13	2,197	2,744	0.5029	1.4624	2	103,636	11.6579	13.3360	19	8.292
						3	268,636		13.3041	20	9.355
						4	475,996		13.3015	20	12.13
						5	818,788		13.3015	21	40.60
very fine	15	3,375	4,096	0.5022	1.4768	2	160,020	12.2274	13.9097	25	29.15
						3	422,412		13.8713	21	26.43
						4	761,100		13.8649	23	43.19
						5	1,338,468		13.8642	100	541.9

* Runtimes measured on an Intel Xeon E3-1245 with 32GB of RAM with MATLAB R2013b.



(a) (b)

Figure 5.18: Optimized structures for the torsion sphere problem in an orthogonal domain. (a) Solution with $N = 5$, $Lvl = 4$ and $N_b = 15,980$. (b) Solution with $N = 13$, $Lvl = 4$ and $N_b = 475,996$.

- Increasing ϕ_F (reducing the radius of the circumference where the load is applied), makes the problem more difficult to approximate numerically. Truss elements, such as those used in the ground structure method cannot handle a moment applied at a single point, and thus as $\phi_F \approx \pi/2$ the optimal volume $V_{opt} \to \infty$ (refer to Equation (5.1)).

- The radius of the analytical closed–form solution (Equation (5.36)), given in Table 5.4, exceeds the boundaries of the discretized domain $2R > L = 1$. Thus, the domain does not fully accommodate the optimal analytical solution. This situation, however, improves with mesh refinement.

- For this problem in particular, the theoretical optimal solution is comprised solely of curved members. A high connectivity level will generate longer straight members, which do not improve the approximation for this analytical solution. Figure 5.19 shows the analytical solution viewed from the poles, along with a fictitious discretization illustrating some members at different connectivity levels: lower level members (shorter) have a better chance at approximating the solution.

5.3.4 Torsion sphere (spherical domain)

Once knowledge is gained that the solution lies in a close–to–spherical domain, the base mesh can be modified to provide a better approximation. The base mesh in this case is a thick hollow sphere. This is not a shell, for it allows the solution to modify its radii if needed. The load is applied at an intermediate radius r_m between the inner and outer radii of the hollow sphere, r_i and r_o respectively as shown in Figure 5.20(a). The domain is represented using spherical coordinates, and is discretized in the θ, ϕ and r directions using N_θ, N_ϕ and N_r elements respectively. A sample mesh discretized with $\{N_\theta, N_\phi, N_r\} = \{30, 14, 2\}$ is shown in Figure 5.20(b), with an axisymmetrical plot given in Figure 5.20(c). The sphere is hollow, and thus a spherical restriction zone with radius equal to the internal radius r_i is used.

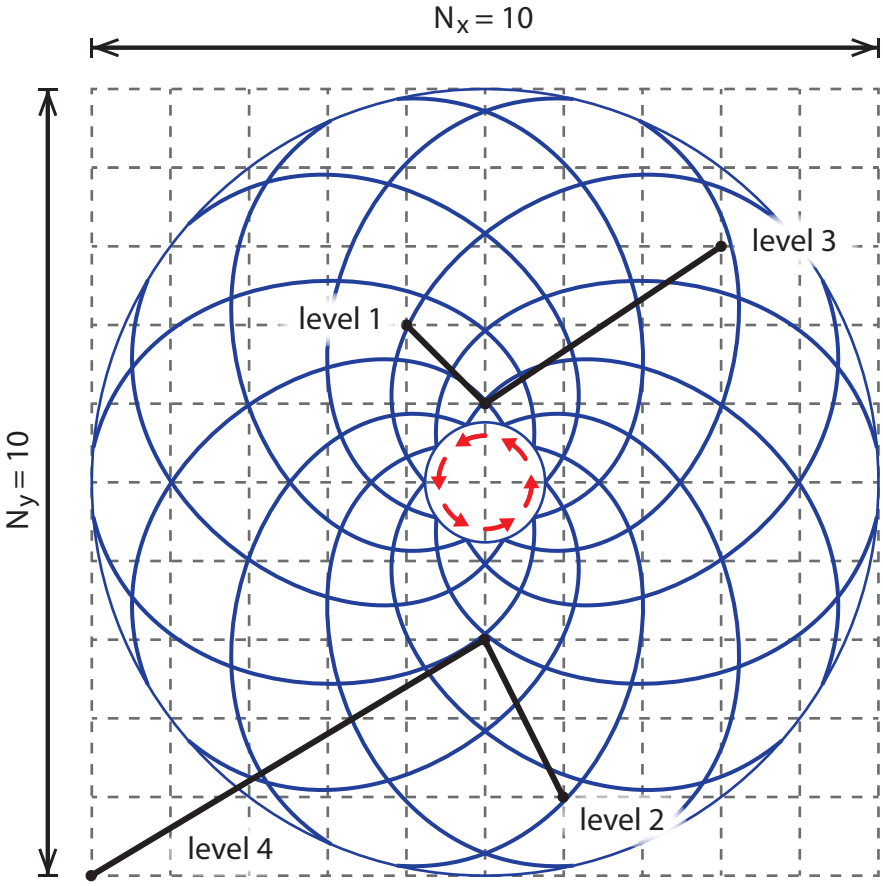


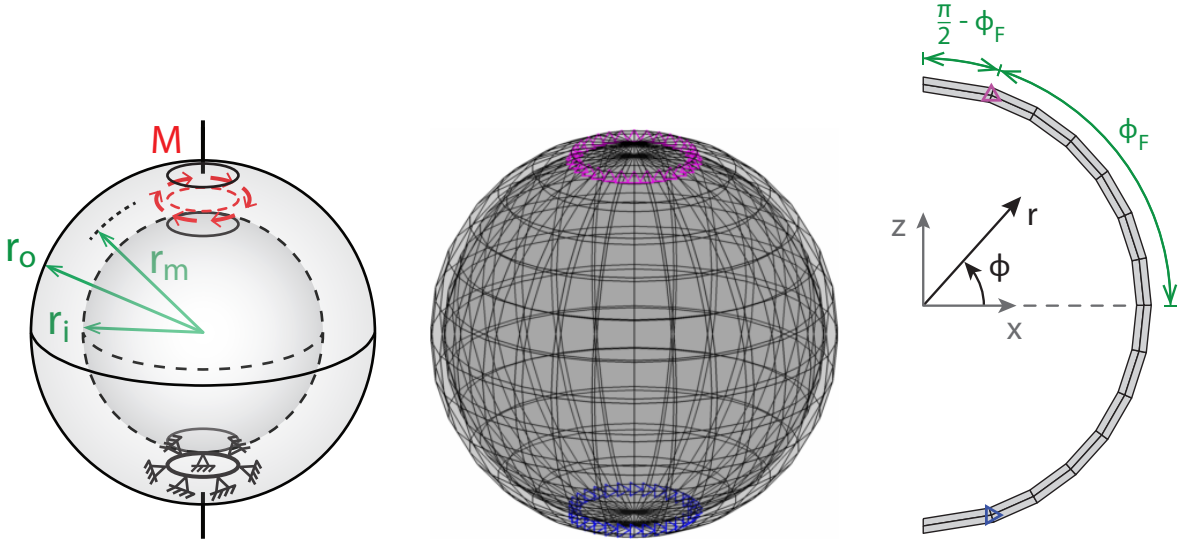
Figure 5.19: Polar view of the analytical closed–form solution for the torsion sphere problem. A fictitious regular discretization with some members is shown to highlight the inability of higher level members to approximate the solution.

If the domain is discretized with a constant spacing on ϕ, then the angle ϕ_F of the applied force is:

$$\phi_F = \frac{\pi}{2}\frac{N_\phi - 2}{N_\phi} \tag{5.37}$$

The optimal volume, given by Equation (5.1), will increase with refinement on the base mesh due to an increase in ϕ_F. Optimal volumes for increasingly refined base meshes with constant spacing are given in Table 5.5.

Ideally, a convergence study should solve the same problem using increasingly refined meshes. To fix the location of the applied loads, the size of the first and last element on the ϕ partition are fixed, and the remaining elements are evenly distributed. Thus making ϕ_F



(a) (b) (c)

Figure 5.20: Torsion sphere problem. (a) Domain definition, loading and supports. (b) Sample mesh with $r_i = 2.9$, $r_m = 3$ and $r_o = 3.1$ discretized with $N_\theta = 30$, $N_\phi = 14$ and $N_r = 2$. (c) Axisymmetric plot of the sample mesh with $\pi/2 - \phi_F = \pi/10$, $r_i = 2.9$, $r_m = 3$ and $r_o = 3.1$ discretized with $N_\phi = 14$ and $N_r = 2$.

Table 5.5: Convergence for a hollow spherical domain with $M = 7$, $r_i = 2.9$, $r_m = 3.0$ and $r_o = 3.1$. The discretization in ϕ is constant; i.e. the angle ϕ_F (and the volume V_{opt}) increases with refinement.

N_θ	N_ϕ	N_r	N_e	N_n	Lvl	N_b	ϕ_F	V_{opt}	Volume V	LP_{iter}	Runtime* [min]
16	8	2	256	342	3	4,308	1.1781	45.2170	48.5088	11	0.008
24	12	2	576	798	3	21,076	1.3090	56.7725	60.2254	10	0.069
32	16	2	1,024	1,446	3	59,780	1.3744	64.8980	67.9621	60	1.801
40	20	2	1,600	2,286	3	121,244	1.4137	71.1785	74.0993	84	5.456
48	24	2	2,304	3,318	3	202,036	1.4399	76.3012	78.9887	32	6.192
56	28	2	3,136	4,542	3	308,844	1.4586	80.6280	83.4937	98	26.11
64	32	2	4,096	5,958	3	425,284	1.4726	84.3738	87.6053	13	13.83
72	36	2	5,184	7,566	3	575,932	1.4835	87.6764	91.4038	110	90.11

* Runtimes measured on an Intel Xeon E3-1245 with 32GB of RAM with MATLAB R2013b.

constant for all discretizations (Figure 5.20(c)), and consequently V_{opt} constant regardless of the refinement in the base mesh. Taking $\pi/2 - \phi_F = \pi/10$, the resulting optimal volume is $V_{opt} = 51.5964$. The resulting optimal volumes for increasingly refined base meshes are given in Table 5.6, with one of these solutions shown in Figure 5.21.

Making ϕ_F constant ensures that the refinement is approximating the same boundary value problem, and thus a smooth convergence to the analytical closed–form solution is obtained. If ϕ_F is variable, then the method is approximating a different problem with each

Table 5.6: Convergence for a hollow spherical domain with $M = 7$, $r_i = 2.9$, $r_m = 3.0$ and $r_o = 3.1$. The discretization in ϕ makes the first and last $\Delta\phi$ equal to $\pi/10$, with the remaining elements evenly distributed; i.e. the angle ϕ_F is constant and equal to $\phi_F = \pi/2 - \pi/10$ for all discretizations. Ground structures are generated with $N_r = 2$ and $Lvl = 3$. The optimal volume is $V_{opt} = 51.5964$.

N_θ	N_ϕ	N_e	N_n	N_b	Volume V	LP_{iter}	Runtime* [min]
20	10	400	546	10,564	54.9909	108	0.261
30	14	840	1,176	38,734	53.6278	10	0.189
40	18	1,440	2,046	96,484	52.8693	11	1.011
50	22	2,200	3,156	178,804	52.4600	12	3.467
60	26	3,120	4,506	283,444	52.2877	110	33.34
70	30	4,200	6,096	415,664	52.1777	12	10.30

* Runtimes measured on an Intel Xeon E3-1245 with 32GB of RAM with MATLAB R2013b.

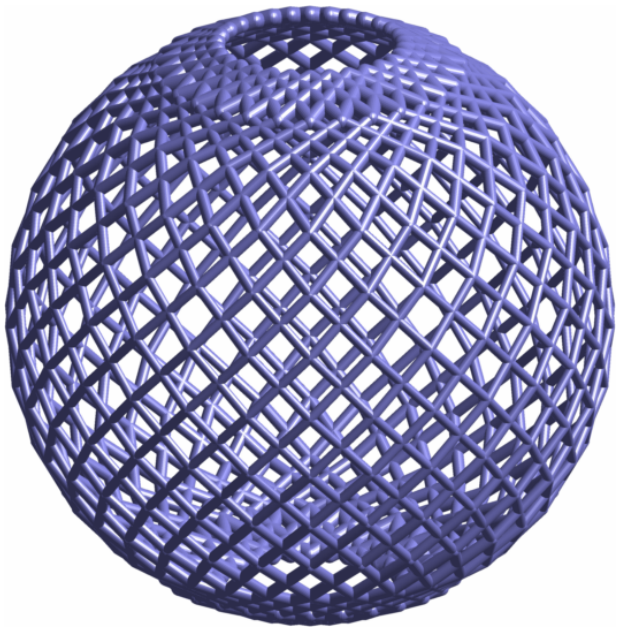


Figure 5.21: Michell's torsion sphere solution obtained for a domain with $r_i = 2.9$, $r_m = 3$ and $r_o = 3.1$. Domain is discretized with $N_\theta = 30$, $N_\phi = 14$ and $N_r = 2$, resulting in $N_e = 840$, $N_n = 1,176$. The ground structure generated with $Lvl = 3$ has $N_b = 38,734$ members, and an optimal volume of $V = 53.6278$.

discretization, and thus convergence is not guaranteed. The torsion sphere problem becomes more difficult to approximate numerically if the radius where the moment couple is applied is small (large ϕ_F). Convergence curves for both situations are shown in Figure 5.22.

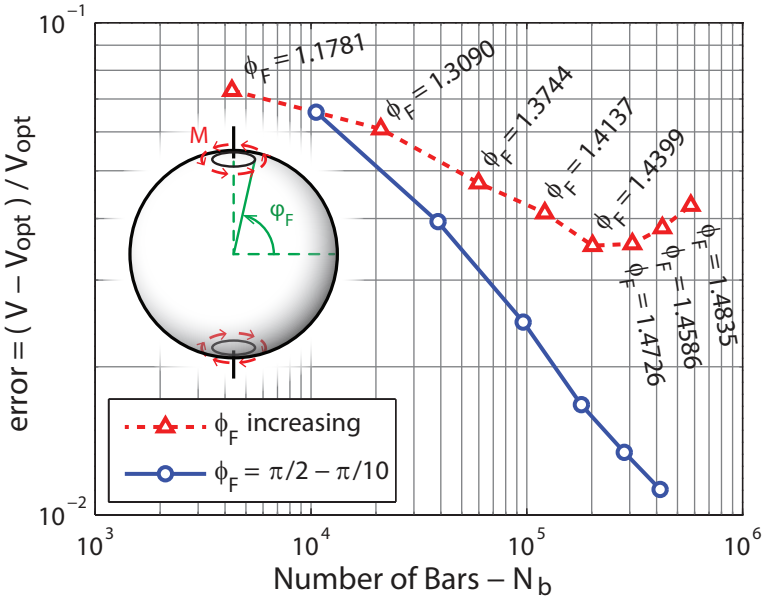


Figure 5.22: Convergence to the optimal solution of increasingly refined spherical base meshes. The case where ϕ_F increases with refinement begins to diverge as $\phi_F \approx \pi/2$. The case where ϕ_F is constant converges as is expected.

5.4 Sample problems

The following problems were selected to showcase features and issues of the method and its implementation (GRAND3). Unless otherwise stated, the stress limit ratio is $\kappa = 1.0$ with $\sigma_T = 1$, the collinear tolerance is $ColTol = 0.999999$, and the plotting cutoff is $Cutoff = 0.005$.

5.4.1 Edge–supported (double) cantilever beam

This problem consists of a three–dimensional *box* domain, fixed at one end on two (opposite) vertical edges, and loaded at the center of the other end (Figure 5.23(a)). The domain has dimensions $L_x = 3$ and $L_y = L_z = 1$, is loaded with $P = 1$, and is discretized with a regular partition in all three dimensions. Examples for a coarse and a fine base mesh are given in Figures 5.23(b) and 5.23(c). The optimal volumes obtained from a series of increasingly refined meshes exhibit convergence to a unique value as expected.

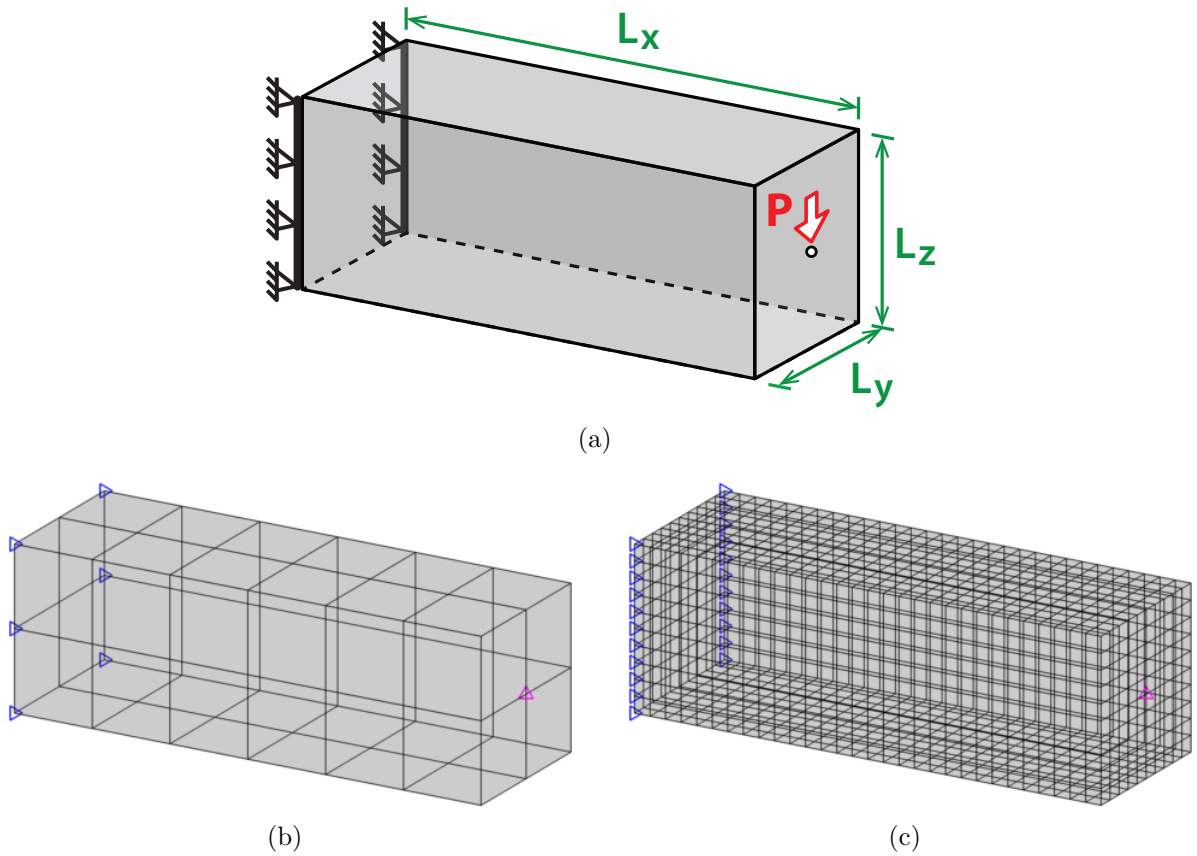
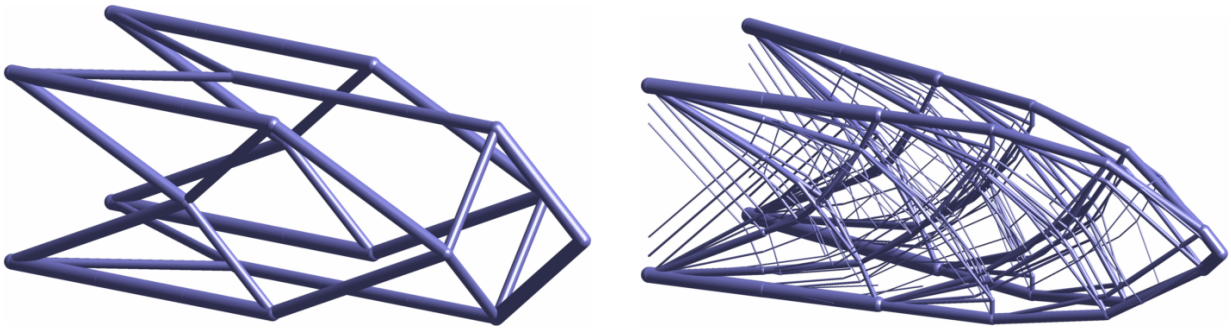


Figure 5.23: Edge–supported double cantilever problem. (a) Domain with loads, boundary conditions and dimensions. (b) Base mesh used to generate a coarse ground structure: $L_x = 3$, $L_y = L_z = 1$ and $P = 1$, discretized with $N_x = 6$ and $N_y = N_z = 2$, resulting in $N_e = 24$ and $N_n = 63$. (c) Base mesh used to generate a fine ground structure: $L_x = 3$, $L_y = L_z = 1$ and $P = 1$, discretized with $N_x = 30$ and $N_y = N_z = 10$, resulting in $N_e = 3,000$ and $N_n = 3,751$.

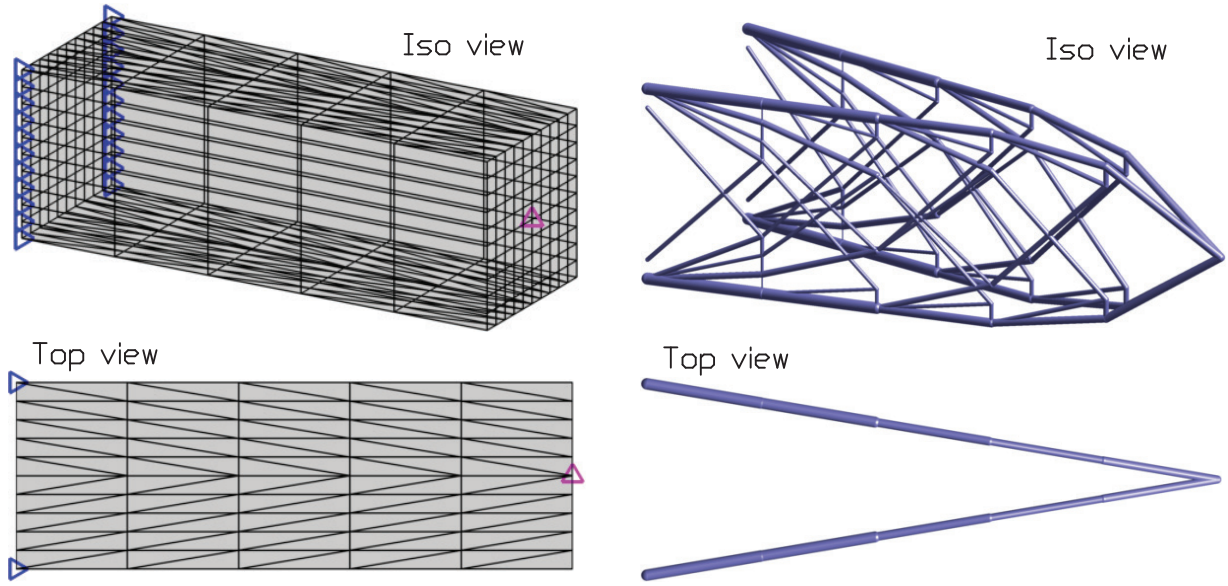
Two sample solutions obtained for the coarse and a fine base meshes are shown in Figures 5.24(a) and 5.24(b). These results hint that optimal closed–form solution consists of two flat cantilever beams (as in Lewiński et al. (1994b)), meeting at the load application point. With this assumption, a new base mesh is created for the problem: this (improved) base mesh allows for perfectly flat cantilevers to develop if the optimal structure requires it, as shown in Figure 5.25(a). The optimal structure obtained for this new base mesh can be seen in Figure 5.25(b), and confirms the flat cantilever hypothesis.

The optimal volumes obtained using regular base meshes show convergence to an absolute



(a) (b)

Figure 5.24: Optimized structures for the edge–supported double cantilever problem. (a) Solution for the coarse base mesh with $N_e = 24$ and $N_n = 63$, using $Lvl = 2$ and $N_b = 962$. (b) Solution for the fine base mesh with $N_e = 3,000$ and $N_n = 3,751$, using $Lvl = 6$ and $N_b = 1,474,218$.



(a) (b)

Figure 5.25: Edge–supported double cantilever problem with improved base mesh discretization. (a) Base mesh used to generate the ground structure: discretized with $N_x = 5$ and $N_y = N_z = 10$, resulting in $N_e = 1,000$ and $N_n = 726$. (b) Solution using the improved base mesh with $Lvl = 6$ and $N_b = 137,877$. Resulting optimal volume is $V = 14.2725$.

optimal in the vicinity of $V_{opt} \approx 13.93$: this value was obtained using the two–dimensional implementation described in Chapter 4 and in Zegard and Paulino (2014a). Convergence data and plots are shown in Table 5.7 and Figure 5.26 respectively.

Table 5.7: Convergence for the three–dimensional double cantilever beam with $L_x = 3$, $L_y = L_z = 1$ and $P = 1$, approximated using a regular–orthogonal mesh.

N_x	N_y	N_z	N_e	N_n	Lvl	N_b	Volume V
6	2	2	24	63	2	962	15.250000
12	4	4	192	325	3	17,604	14.680556
18	6	6	648	931	4	112,374	14.458478
24	8	8	1,536	2,025	5	499,112	14.255496
30	10	10	3,000	3,751	6	1,474,218	14.123627
36	12	12	5,184	6,253	7	4,078,236	14.054298

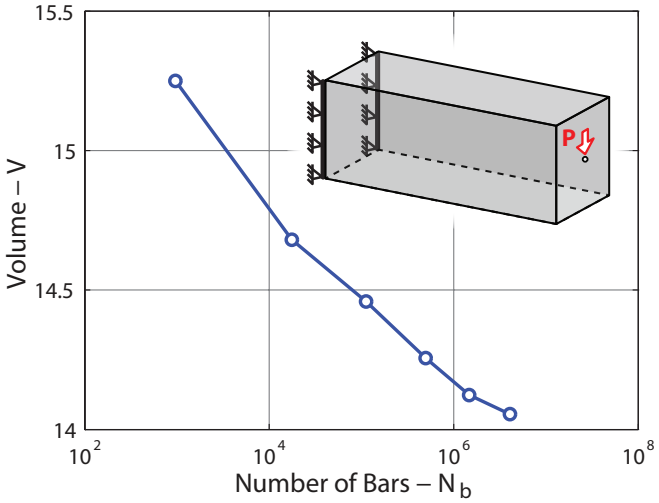
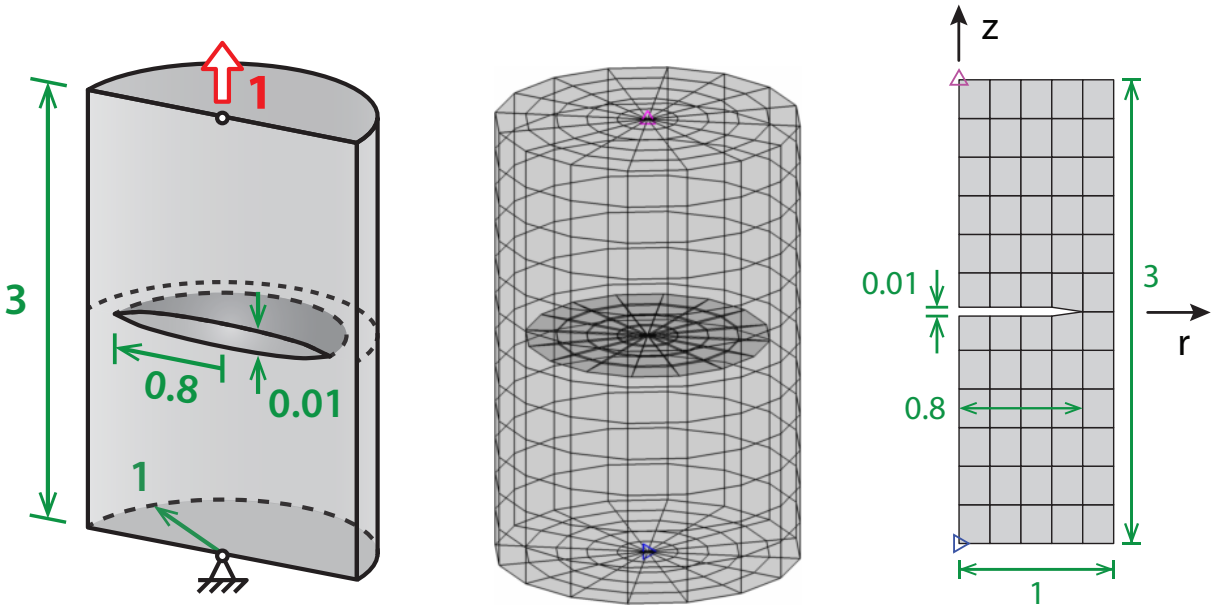


Figure 5.26: Convergence of the optimal volume for the edge–supported double cantilever problem, for a set of increasingly refined ground structures.

5.4.2 Diamond problem

The diamond problem is a cylindrical domain with a coin–shaped (or disc) discontinuity in the middle. This problem shows the capability of the method to find optimal load paths for problems that include discontinuities, imperfections or barriers in their structure. The domain has a vertical load along it's z axis, as shown in Figure 5.27(a). The domain is meshed as in Figures 5.27(b) and 5.27(c). The ground structure is generated for a $Lvl = 3$ connectivity, resulting in $N_b = 109,820$. The restriction zone is a disc primitive in the coin–shaped discontinuity. The resulting optimal structure resembles a *diamond*: the members fan away from the axis to evade the discontinuity. At the point of maximum width, it creates



(a) (b) (c)

Figure 5.27: Diamond problem: Vertically loaded cylinder with a coin–shaped discontinuity. (a) Half–domain with loads, boundary conditions and dimensions. (b) Base mesh used to generate the ground structure: $N_z = 12$, $N_\theta = 16$ and $N_r = 5$. (c) Axisymmetric plot of the base mesh with $N_z = 12$ and $N_r = 5$.

a *strong ring* with the purpose of shifting the member orientation back into the support as shown in Figure 5.28.

5.4.3 Cup problem (spider)

This problem consists of an inverted cup–shaped domain, loaded vertically in the interior. Figure 5.29(a) shows the half–domain with loads, boundary conditions and dimensions. The domain is discretized using cylindrical coordinates, and the restriction zone is a single rod primitive in the interior of the cup. The resulting mesh is shown in Figure 5.29(b), with a axisymmetric view given in Figure 5.29(c). The solution to this problem is shown in Figure 5.30(a).

The optimal solution at the top (Figure 5.30(b)) is degenerate; the solution lies on a facet of the feasible domain. This is not an issue of the ground structure method, but rather

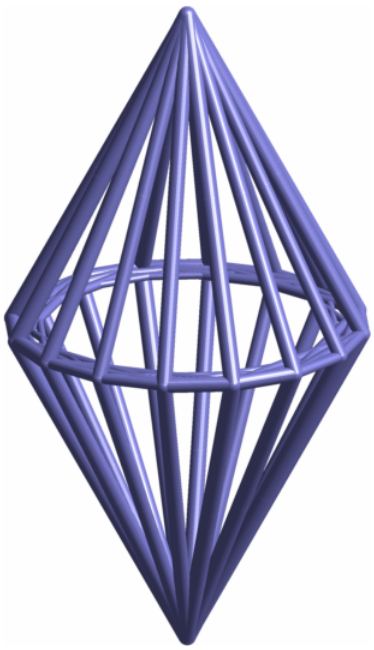
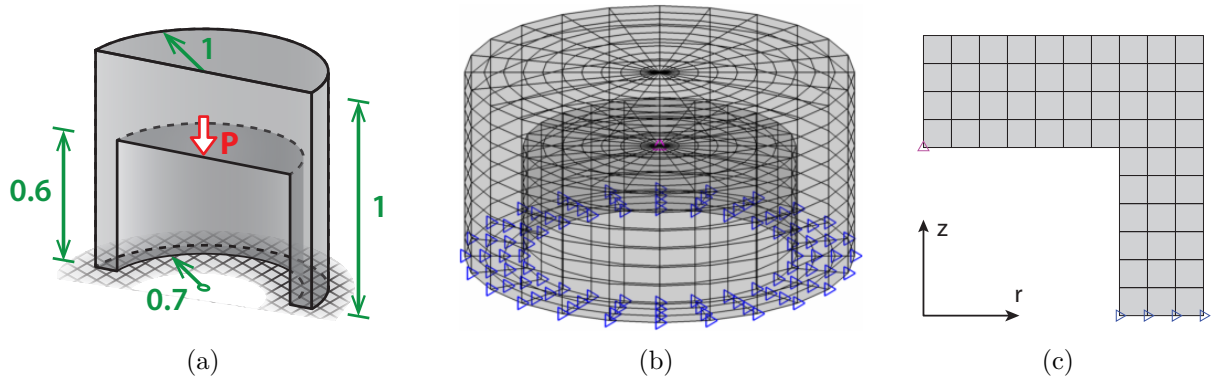


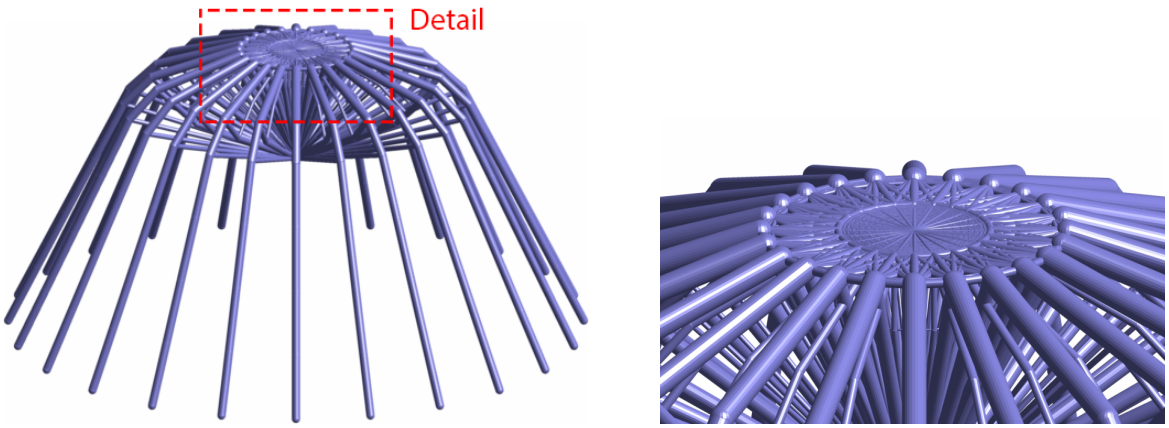
Figure 5.28: Optimal solution obtained for the diamond problem using $N_z = 12$, $N_\theta = 16$, $N_r = 5$, $Lvl = 3$ and $N_b = 109,820$. The optimized volume is $V = 4.7067$.



(a) (b) (c)

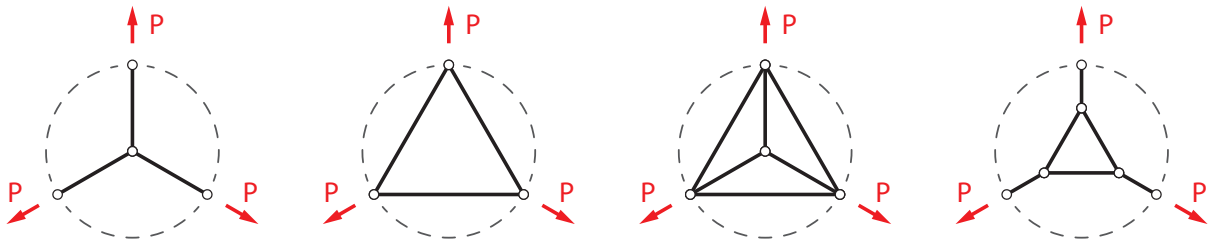
Figure 5.29: Vertically loaded inverted cup problem. (a) Half–domain with loads, boundary conditions and dimensions. (b) Base mesh used to generate the ground structure. (c) Axisymmetric plot of the base mesh.

a characteristic of the problem: it does not have a unique solution (Rozvany, 1997a). This situation can be better understood with a simple 3 force problem (Mazurek et al., 2011): the optimal structure to carry 3 forces evenly distributed on a circle of radius R is not unique, as shown in Figure 5.31. For the topology in Figure 5.31(a), the force in each member is



(a) (b)

Figure 5.30: Optimal solution to the inverted cup problem. Problem's parameters are $N_e = 1,392$, $N_n = 1,781$, $Lvl = 3$ and $N_b = 168,436$, resulting in an optimal volume of $V = 2.9384$. (a) Plot of the optimal structure using GRAND3. (b) Detail of the optimal structure.



(a) (b) (c) (d)

Figure 5.31: Structural optimization problem with a non–unique (degenerate) solution: (a), (b), (c) and (d) are all optimal topologies.

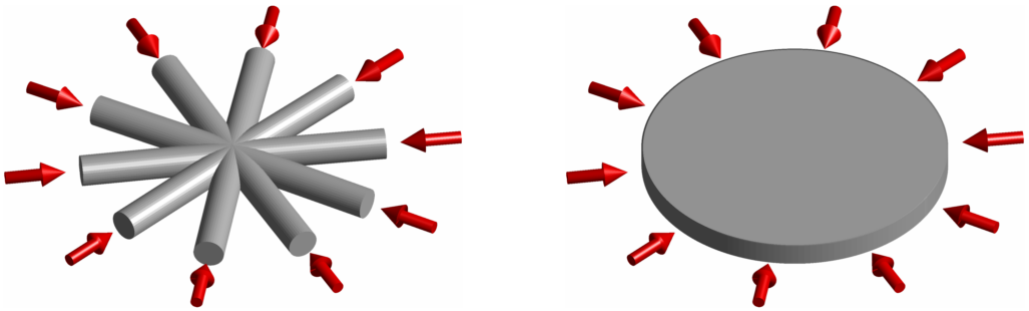
$N_i = P$, and the length of the members is $L_i = R$. Thus, the optimal volume is:

$$V_{opt1} = 3\left(\frac{P}{\sigma_T}\right)(R) = 3\frac{PR}{\sigma_T}$$

Repeating the analysis for the topology in Figure 5.31(b), the force in each member is $N_i = P/\sqrt{3}$, and the member's length is $L_i = \sqrt{3}R$. Thus, the optimal volume is:

$$V_{opt2} = 3\left(\frac{P/\sqrt{3}}{\sigma_T}\right)\left(\sqrt{3}R\right) = 3\frac{PR}{\sigma_T}$$

In fact, any combination of the previous two cases, as in Figures 5.31(c) and 5.31(d) for example, will result in the same optimal volume. In practice, when facing an optimal structure with a degenerate solution like in Figure 5.30(a), the engineer can decide the final topology. Figures 5.32(a) and 5.32(b) are examples of possible optimal topologies for the problem in Figure 5.30(b).



(a) (b)

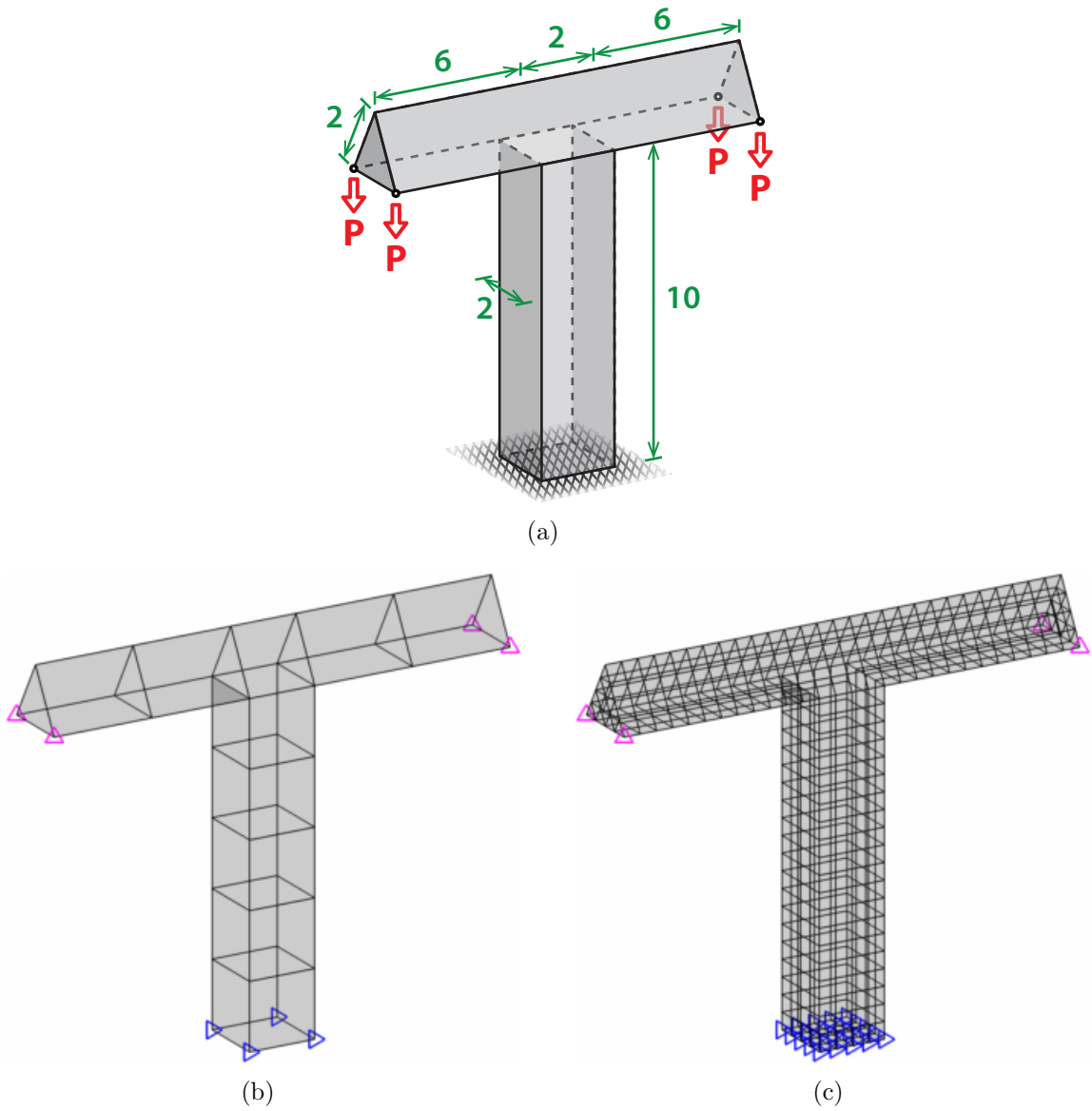
Figure 5.32: Options of topologies for a degenerate problem: (a) Spoke and hub option. (b) Slab option.

5.4.4 Crane problem

The crane (or tower) problem was introduced by Smith (1998), in whose work the domain had to be partitioned into regions in a specialized CAD system. A three–dimensional ground structure was then generated within these regions.

The approach presented here (and implemented in GRAND3), requires no subdivision of the domain, but does require the definition of *restriction zones*. Domains are often procedurally generated using geometric primitives. Because the restriction zone is typically a subset of the primitives used to construct the domain, the additional work required is comparatively small.

The domain in Figure 5.33(a) is discretized with two different degrees of refinement as in Figures 5.33(b) and 5.33(c). The restriction zone is the union of two *boxes* under both of the tower's arms. The optimized ground structures for each case are shown in Figures 5.34(a) and



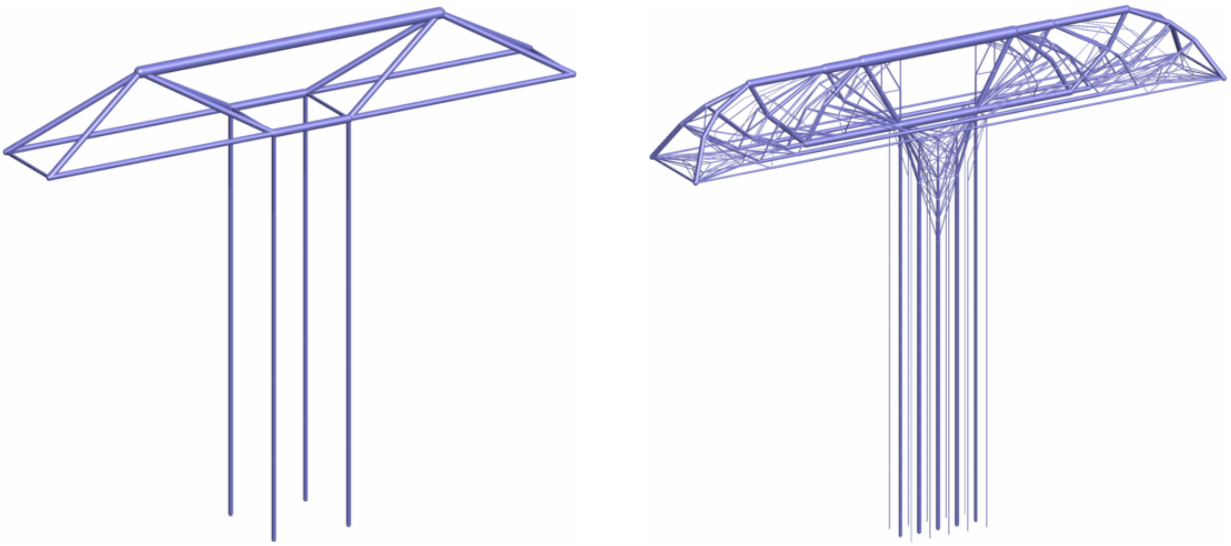
(a)

(b)

(c)

Figure 5.33: Crane problem: Tower with arms loaded at four points. (a) Domain with loads, boundary conditions and dimensions. (b) Base mesh used to generate a coarse ground structure: $N_e = 10$ and $N_n = 38$. (c) Base mesh used to generate a fine ground structure: $N_e = 768$ and $N_n = 935$.

5.34(b). This example showcases the ability of the method (and implementation) to provide solutions with different levels of detail. Highly detailed solutions provide information into the optimal load transfer mechanism, while coarse solutions are more likely to be developed into real structures.



(a) (b)

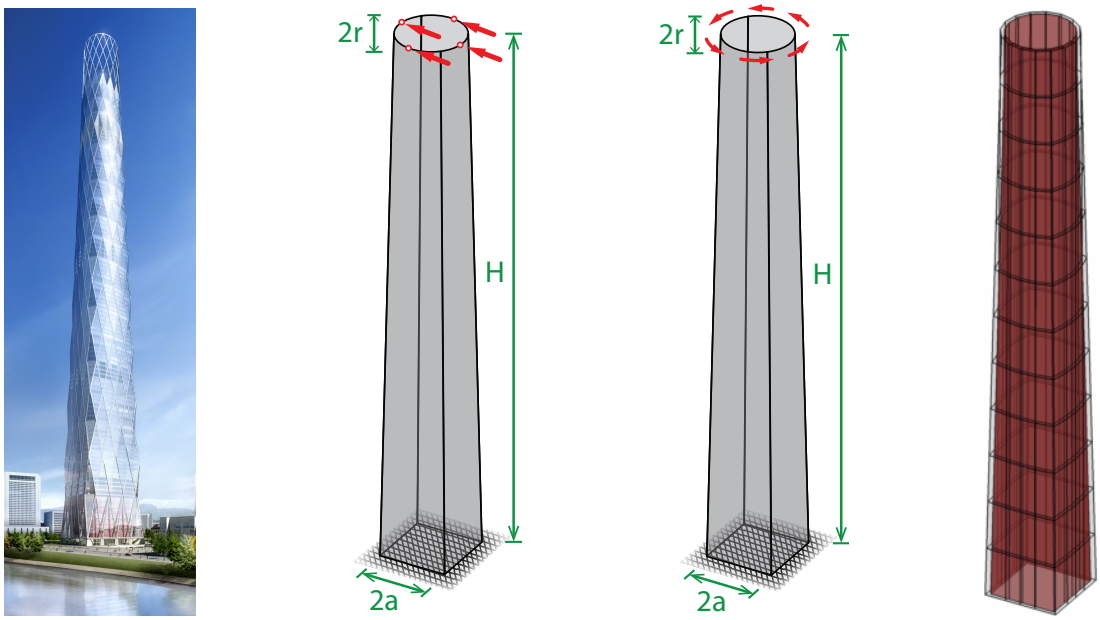
Figure 5.34: Optimized ground structures for the crane problem. (a) Solution for the coarse base mesh: $Lvl = 3$ and $N_b = 315$. (b) Solution for the fine base mesh: $Lvl = 3$ and $N_b = 47,076$; the plotting cutoff is $Cutoff = 0.002$ to prevent members from ending mid–air.

5.4.5 Lotte tower (Seoul, South Korea)

The Lotte tower is a shell–like domain (no thickness) that is square at the base and circular at the top. This problem was inspired by the design competition of the same name by Skidmore, Owings & Merill LLP (Figure 5.35(a)), and has been previously optimized using a density–based optimization approach by Stromberg et al. (2010). In the present example, the domain is optimized for two loading scenarios: lateral load at the tip acting at 4 points (Figure 5.35(b)), and a torsional load distributed over the top circumference (Figure 5.35(c)).

The domain's height is $H = 80$, the square at the base has sides $2a = 10$, and the circle at the top has a diameter of $2r = 10$. The domain is discretized using quadratic surface elements; partitioned into $N_z = 12$ elements along its height, and $N_\theta = 16$ elements around, resulting in $N_e = 192$ and $N_n = 208$. The restriction zone is a surface primitive set–back from the interior of the domain by a small spacing, as shown in Figure 5.35(d).

The ground structure is generated for a $Lvl = 5$ connectivity, resulting in $N_b = 4,100$



(a) (b) (c) (d)

Figure 5.35: Lotte tower problem: (a) Rendering of the Lotte tower [© Skidmore, Owings & Merrill LLP]. (b) Domain definition, loading and boundary conditions for the laterally loaded tower. (c) Domain definition, loading and boundary conditions for the torsionally loaded tower. (d) Base mesh for the ground structure generation, with the restriction surface also shown.

potential members. The optimized ground structures for the lateral loading and torsional loading cases are shown in Figures 5.36(a) and 5.36(b) respectively.

The lateral loading causes part of the resulting topology to work as a *web* (front view in Figure 5.36(a)); developing an arrangement similar to an optimal cantilever, and analogous to the two–dimensional problem in Section 4.3.1 (Figure 4.12). The other side of the solution (side view in Figure 5.36(a)) acts as a *flange*, transferring the load in tension/compression away from the *web* and to the foundation supports. It is understood that buildings are subject to lateral loads from any directions and therefore 4-axis symmetry in this design could be enforced.

The optimized tower for torsion (Figure 5.36(b)) results in a diagrid pattern, similar to those used in high–rise buildings. While the torsional stiffness of structural systems is impor-

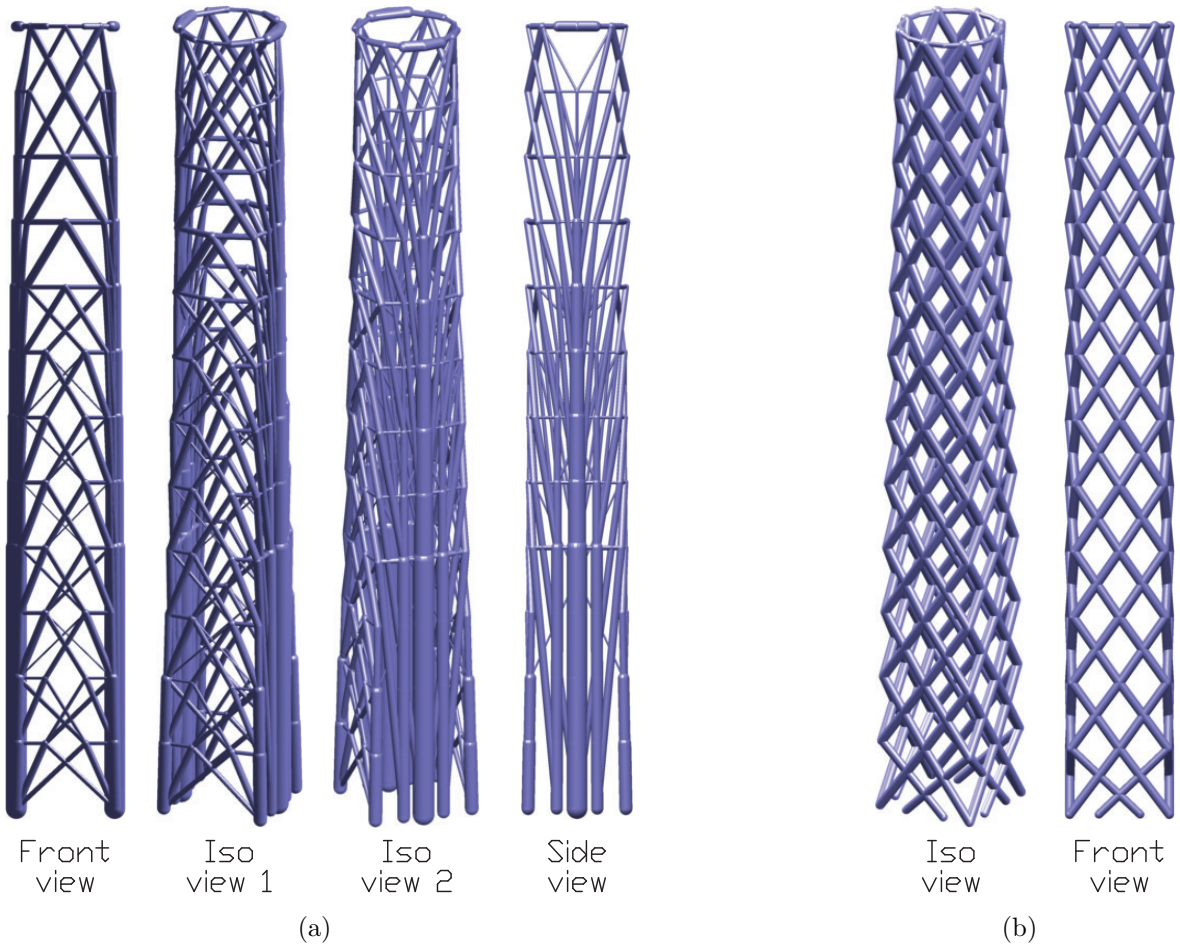


Figure 5.36: Lotte tower problem: (a) Optimized ground structure for a lateral loading at the top. (b) Optimized ground structure for a torsional load at the top.

tant in tall building design, a combination of lateral and torsional stiffness requirements will likely result in a better design. Thus, it can be anticipated that a more rigorous optimization process will likely result in a hybrid of the designs in Figures 5.36(a) and 5.36(b).

5.5 Conclusions

The ground structure generation and analysis methodology presented in the previous chapter is successfully extended to three–dimensional space. Concavities and holes in the domain are addressed using three–dimensional collision primitives, extending the approach proposed

in Chapter 4. Collision primitives for a variety of shapes are explained or developed, among which the *surface primitive* offers a viable alternative in the eventual case that no other primitive is adequate.

The method is verified and benchmarked using three problems for which the solutions are known, with the method converging to the analytical solution. Features and details fot he convergence process are explored in detail. Additional problems showcase the capabilities of the method to address complicated domains, and to generate structures with various levels of detail. The ability to handle domains other than *boxes* has resulted in innovative bio–inspired and natural designs, as is the case with the "Diamond problem" and the "Cup domain or spider" (Figures 5.28 and 5.30(a) respectively).

Limited computational power, and the rapid scaling of the complexity in three–dimensional space, make it necessary for the user to employ *engineering judgement* in order to obtain reasonably *good* solutions for applied problems. A *good* solution is such that it can be obtained within reasonable computer time, is detailed enough, and can be manufactured. The "Lotte tower" problem loaded under torsion (Section 5.4.5), for example, results in a diagrid structure that is both; beautiful and feasible.

Chapter 6

Additive manufacturing of optimal structures

The field of structural optimization has developed for over a century (Topping, 1983; Rozvany, 2009; Deaton and Grandhi, 2013). However, the ability to manufacture these structures lags behind our ability to design and optimize them. Recently, *additive manufacturing* (colloquially known as *3D printing*), positions itself as the missing link towards a fully integrated optimal structural design: the materialization.

Additive manufacturing opens the possibility to overcome limits currently imposed by conventional manufacturing techniques. There is a large variety of additive manufacturing technologies. However, the steep cost and size of these machines indirectly restricted these to large industry and research installations. Recently, there has been a push to bring these technologies to the consumer and small industry (Jones et al., 2011). Five of the most common additive manufacturing technologies are:

1. Fused Deposition Modeling (FDM);
2. Stereolithography (SLA);
3. PolyJet;
4. Selective Laser Sintering (SLS);
5. Selective Laser Melting (SLM).

In FDM (Crump, 1992), layers are built by extruding material, joining it to previously built layers in the process. In SLA (also referred as LS), consecutive layers of photopolymer liquid are cured by a UV laser or similar (Hull, 1986). PolyJet technology is similar to SLA, except that the photopolymer is jetted in thin layers onto the model and rapidly cured by a UV light. SLS and SLM fuse material powder in layers, with each consecutive layer commencing by depositing a new layer of powder (Meiners et al., 1998; Deckard, 1989). The main difference between SLS and SLM is whether the material gets fully melted by the laser or not. The cost reduction and improved reliability of Fused Deposition Modeling (FDM) have generated increased awareness and widespread use of 3D printing (Crump, 1992; Jones et al., 2011; Wittbrodt et al., 2013).

This chapter aims to provide a simple and effective procedure for the last step in the design of optimal structures: the manufacture. Three different types of optimal structures are addressed:

- Two–dimensional ground structure optimization
- Three–dimensional ground structure optimization
- Density–based topology optimization

The goal in all three cases is the same—to generate three–dimensional data in a format that can be used for additive manufacturing. The workflow and techniques presented here apply to most (if not all) of the additive manufacturing technologies. Work combining topology optimization techniques (density–based methods mostly) and additive manufacturing do exist (Brackett et al., 2011; Dewhurst and Srithongchai, 2005; Dewhurst and Taggart, 2009; Meisel et al., 2013; Reinhart and Teufelhart, 2011; Rezaie et al., 2013; Sundararajan, 2011; Villanueva and Maute, 2014). However, the use of these technologies is still novel, and further research by the scientific community is required to streamline the process and extend it to a variety of situations.

6.1 Refinement of intermediate values in density–based topology optimization

Topology optimization is making progress with a number of different techniques or approaches: density–based, ground structure, truss layout, level set, phase field, evolutionary and several others (Topping, 1983; Bendsøe and Sigmund, 2003; Deaton and Grandhi, 2013; Sigmund and Maute, 2013). All these try to answer the same question: What is the best distribution of material within a prescribed domain?

This section briefly reviews the theory and concepts behind density–based methods. The topology optimization problem can be solved by two approaches: the *nested* method, where the structural equilibrium equations are assumed to be satisfied for each optimization step, i.e. alternating design update and analysis steps; and the *simultaneous* method, that concurrently optimizes for the solution of the structural equilibrium and design variables (Christensen and Klarbring, 2009). This work focuses on the *nested* formulation, with design updates guided by gradient information.

The topology optimization problem may be formulated as follows: find the material distribution that minimizes some objective function J, subject to a volume constraint $g_0 \leq 0$, and possibly N_c additional constraints $g_i \leq 0,\ i = 1 \dots N_c$. The material distribution is described by $\chi(\mathbf{x})$, that can take either the value 0 for void, or 1 for solid, at any point $\mathbf{x}$ in the design domain Ω. Thus, the optimization problem becomes:

$$\begin{aligned} \min_{\chi} \quad & J(\chi, \mathbf{u}(\chi)) \\ \text{s.t.} \quad & g_0(\chi) = \textstyle\int_\Omega \chi(\mathbf{x})\, dV - (f)(V_0) \leq 0 \\ & g_i(\chi, \mathbf{u}(\chi)) \leq 0 \qquad i = 1 \dots N_c \\ & \chi(\mathbf{x}) = \{0, 1\} \qquad \forall\, \mathbf{x} \in \Omega\,, \end{aligned} \tag{6.1}$$

where $f = [0, 1]$ is the specified volume constraint, V_0 is the volume of the design domain,

and $\mathbf{u}$ is the displacement field that satisfies the equations of elasticity[1].

The continuous problem described in Equation (6.1) is typically solved by discretizing the domain Ω into a large number of finite elements. In its simplest form, each finite element has an associated density value, constant within the element[2]. Therefore, the density distribution is represented in a piecewise constant fashion throughout the domain. The discretized optimization problem is then:

$$\begin{aligned} \min_{\boldsymbol{\chi}} \quad & J\left(\boldsymbol{\chi}, \mathbf{u}\left(\boldsymbol{\chi}\right)\right) \\ \text{s.t.} \quad & g_0\left(\chi\right) = \sum_j^{N_e} \chi_j v_j - \left(f\right)\left(V_0\right) \leq 0 \\ & g_i\left(\boldsymbol{\chi}, \mathbf{u}\left(\boldsymbol{\chi}\right)\right) \leq 0 \qquad i = 1 \ldots N_c \\ & \chi_j = \{0, 1\} \qquad j = 1 \ldots N_e \\ \text{with} \quad & \mathbf{K}\left(\boldsymbol{\chi}\right)\mathbf{u} = \mathbf{f}\,, \end{aligned} \tag{6.2}$$

where v_i is the volume associated with the i-th element, for all N_e elements in the domain. In addition, there is an implicit relationship between the design variables $\boldsymbol{\chi}$ and the displacements $\mathbf{u}$; where $\mathbf{K}$ is the (assembled) global stiffness matrix in accordance to $\boldsymbol{\chi}$, and $\mathbf{f}$ is the nodal force vector.

[1]Consider an isotropic elastic material in Ω: the static theory of linear elasticity requires that:

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma}\left(\mathbf{u}\right) + \mathbf{b} = 0 \quad & \text{in } \Omega \\ \mathbf{u}|_{\Gamma_1} = \mathbf{u_0} & \\ \boldsymbol{\sigma}\left(\mathbf{u}\right) \cdot \mathbf{n}|_{\Gamma_2} = \mathbf{t}\,, & \end{aligned}$$

with $\mathbf{u}$ being the displacement, $\mathbf{b}$ the body force, $\boldsymbol{\sigma}$ the stress tensor, $\mathbf{u_0}$ the prescribed displacements in the boundary Γ_1, and $\mathbf{t}$ the prescribed traction in the boundary Γ_2 with outer normal $\mathbf{n}$. The boundary is defined everywhere, i.e. $\partial\Omega = \Gamma_1 \cup \Gamma_2$ and $\Gamma_1 \cap \Gamma_2 = \emptyset$. The stress tensor $\boldsymbol{\sigma}\left(\mathbf{u}\right)$ is defined as:

$$\boldsymbol{\sigma}\left(\mathbf{u}\right) = 2\mu\boldsymbol{\epsilon}\left(\mathbf{u}\right) + \lambda \mathrm{tr}\left\{\boldsymbol{\epsilon}\left(\mathbf{u}\right)\right\}\mathbf{I}\,,$$

where the positive constants μ and λ are called the Lamé parameters and are related to the elastic material properties. The strains $\boldsymbol{\epsilon}$ correspond to the symmetrical part of the displacement gradient:

$$\boldsymbol{\epsilon} = \frac{1}{2}\left\{\nabla\mathbf{u} + \left(\nabla\mathbf{u}\right)^T\right\}$$

[2]Alternative approaches include assigning the density variables to the nodes (Matsui and Terada, 2004), to a finer embedded mesh (Nguyen et al., 2009), among others.

In the discretized optimization problem (Equation (6.2)), the design variables χ_i can only take discrete values: 0 or 1. This discrete problem is computationally difficult to solve, especially considering the large number of design variables involved. By allowing the density variables to take continuous values between 0 and 1, enables the use of gradient–based optimization algorithms. Gradient–based optimization has rapid convergence, even with the large number of variables involved. The structural equilibrium will encounter computational difficulties with the void elements. It is thus common to replace the void by an Erzats representation instead, i.e. the void is modeled by a very weak material (compared to the solid). Hence, introducing a lower limit ρ_{min} in the density variable, then the problem becomes:

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & J\left(\boldsymbol{\rho}, \mathbf{u}\left(\boldsymbol{\rho}\right)\right) \\ \text{s.t.} \quad & \textstyle\sum_j^{N_e} \rho_j v_j - \left(f\right)\left(V_0\right) \leq 0 \\ & g_i\left(\boldsymbol{\rho}, \mathbf{u}\left(\boldsymbol{\rho}\right)\right) \leq 0 \qquad i = 1 \ldots N_c \\ & 0 \lesssim \rho_{min} \leq \rho_j \leq 1 \qquad j = 1 \ldots N_e \\ \text{with} \quad & \mathbf{K}\left(\boldsymbol{\rho}\right) \mathbf{u} = \mathbf{f}\,, \end{aligned} \tag{6.3}$$

where $\mathbf{K}$ is the assembled on a per–element basis, with stiffness distributions in accordance to the now continuous density variable ρ_i.

The material properties for solid and void (and intermediate values) are defined by a *material interpolation* scheme. Using the SIMP (Solid Isotropic Material with Penalization), or power–law, proposed by Bendsøe (1989) and Zhou and Rozvany (1991), the material property associated with the i-th element is defined as:

$$E_i\left(\rho_i\right) = \rho_i^p E_0 \qquad \text{with} \quad p \geq 1\,, \tag{6.4}$$

where p is the penalization parameter, and E_0 is the Young's modulus of the solid[3]. The

[3]Alternative interpolation schemes have been developed to address some of the issues associated with the SIMP (Bendsøe and Sigmund, 1999; Stolpe and Svanberg, 2001; Bruns, 2005; Dzierżanowski, 2012).

case of $p = 1$ corresponds to the *variable thickness sheet* problem, that for compliance minimization is known to be convex and with a unique solution (Petersson, 1999). Penalization values $p > 1$ will cause the problem to have multiple local minima, but penalizes the intermediate density values. High values for the penalization p will result in a solution close to solid—void (or 0—1), but will likely converge to a local minimum. The *modified SIMP* rescales the standard SIMP so as to remove the lower limit on the density variable (Sigmund, 2007):

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & J\left(\boldsymbol{\rho}, \mathbf{u}\left(\boldsymbol{\rho}\right)\right) \\ \text{s.t.} \quad & \textstyle\sum_j^{N_e} \rho_j v_j - \left(f\right)\left(V_0\right) \leq 0 \\ & g_i\left(\boldsymbol{\rho}, \mathbf{u}\left(\boldsymbol{\rho}\right)\right) \leq 0 \qquad i = 1 \ldots N_c \\ & 0 \leq \rho_j \leq 1 \qquad j = 1 \ldots N_e \\ & E_k\left(\rho_k\right) = E_{min} + \rho_k^p \left(E_0 - E_{min}\right) \qquad k = 1 \ldots N_e \\ \text{with} \quad & \mathbf{K}\left(\boldsymbol{\rho}\right)\mathbf{u} = \mathbf{f}\,, \end{aligned} \tag{6.5}$$

where $E_{min} > 0$ is the modulus of elasticity of the Ersatz material (very weak material used to represent the void). The domain may also include *passive* elements (or associated design variables). Passive elements can be prescribed to be void or solid, and are referred to as *passive–void* and *passive–solid* respectively. The density variables to be optimized are called *active*. The number of active, passive–void and passive–solid variables are N_a, N_{pv} and N_{ps} respectively, with $N_e = N_a + N_{pv} + N_{ps}$. It may be desirable to specify a volume fraction $f^\star$ for the design domain only (i.e. the active region). This volume fraction $f^\star$ is related to the total volume fraction f by:

$$f = \frac{f^\star \sum_j^{N_a} v_j + \sum_k^{N_{ps}} v_k}{\sum_j^{N_a} v_j + \sum_k^{N_{ps}} v_k + \sum_l^{N_{pv}} v_l} = \frac{f^\star \sum_j^{N_a} v_j + \sum_k^{N_{ps}} v_k}{\sum_i^{N_e} v_i} = \frac{f^\star \sum_j^{N_a} v_j + \sum_k^{N_{ps}} v_k}{V_0} \tag{6.6}$$

The continuous nature of the density variable ρ is not physical, and a solid—void (or 0—1) solution is desired: some interpretation process is required. The solid boundary can be defined by a cutoff (or threshold) value: densities $\rho > \mathit{cutoff} \approx 0.5$ are considered to

be solid. The interpreted solution is no longer optimal and the volume constraint may be violated. More sophisticated approaches can ensure that these variations are not significant (Sigmund, 2007; Xu et al., 2010). Nonetheless, taking $cutoff \approx 0.5$ does yield reasonably good results, specially if the results have a small amount of intermediate densities.

The topology optimization problem in Equation (6.1) is not well–posed: the objective can always be decreased by increasingly more and smaller holes throughout the entire domain (Sigmund and Petersson, 1998). Similarly, for the discretized problem in Equations (6.2), (6.3) and (6.5), a refinement of the finite element mesh will typically cause more holes to appear in the solution, i.e. the solution is mesh–dependent. For the problem to be well–posed, the problem needs to be relaxed (Bendsøe and Kikuchi, 1988) or restricted (the latter being more popular in current approaches).

There are a variety of restriction methods for SIMP problems: perimeter control (Ambrosio and Buttazzo, 1993; Haber et al., 1996), sensitivity filter (Sigmund, 1997), density filter (Bruns and Tortorelli, 2001; Bourdin, 2001), projection filter (Guest et al., 2004; Sigmund, 2007; Xu et al., 2010; Wang et al., 2011). The present work uses the density filter, which defines the elements' physical densities $\overline{\boldsymbol{\rho}}$ as the weighted average of the design variables $\boldsymbol{\rho}$. The weighting function is local in nature and operates over the variables in a neighborhood of radius r_{min}. The SIMP with density filtering is a class of the so called *two–field SIMP*, because it makes use of a design variable $\boldsymbol{\rho}$ and a density variable $\overline{\boldsymbol{\rho}}$ field (Sigmund and Maute, 2013). This filter can also be viewed as a convolution operation over the design variables, and implicitly controls the minimum length–scale of the resulting topology.

The density filter has the downside of encouraging a smooth transition region between solid and void. This is also true for other filters as well. There are a number of approaches to reduce the amount of intermediate densities in the transition region, mainly projection and continuation (Allaire and Kohn, 1993; Allaire and Francfort, 1993; Sigmund and Petersson, 1998).

6.1.1 Filters for density–based topology optimization in 3D

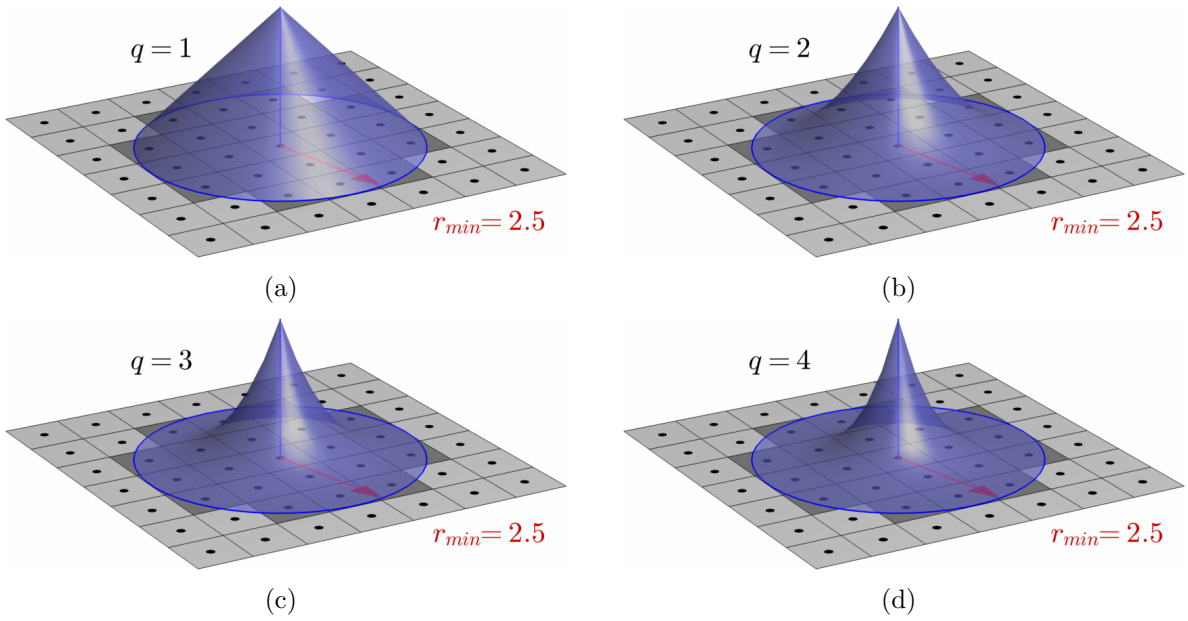


Figure 6.1: Normalized convolution (weighting) functions for two–dimensional filters in a sample patch from a regular and orthogonal mesh. Shaded elements have weights different than zero. (a) Linear filter. (b) Quadratic filter. (c) Cubic filter. (d) Quartic filter.

Linear filters are often used in two–dimensional density–based topology optimization (Sigmund, 2001; Bendsøe and Sigmund, 2003; Andreassen et al., 2011; Liu and Tovar, 2014). However, it has been pointed out that linear filters do not clearly define the solid—void boundary, as they create a smooth transition region between solid and void. The filter is necessary to make the problem well–posed and prevent checkerboard patches in the solution (Díaz and Sigmund, 1995; Sigmund and Petersson, 1998). The checkerboard problem is a numerical artifact common in meshes with *traditional elements* (e.g. triangles, quadrilaterals, hexahedra, wedges to name a few), where the resulting topology is artificially stiff when exhibiting a checkerboard solid—void pattern. In practice, the filter implicitly introduces a minimum length scale of the details in the topology. The physical density of an element $\overline{\boldsymbol{\rho}}$

is a weighted average of the design variables $\boldsymbol{\rho}$ and their associated volumes v as:

$$\begin{aligned} \overline{\boldsymbol{\rho}} &= \mathbf{H}\boldsymbol{\rho} \\ \text{with} \quad \mathbf{H}_{ij} &= \frac{h\left(i,j\right)v_j}{\sum_k^{N_e} h\left(i,k\right)v_k} \\ h\left(i,j\right) &= \begin{cases} \left[r_{min} - \text{dist}\left(i,j\right)\right]^q & \text{for} \quad r_{min} - \text{dist}\left(i,j\right) > 0 \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{6.7}$$

where matrix $\mathbf{H}$ contains the weights relating the design and density variables. The operator $\text{dist}\left(i,j\right)$ is defined as the distance between density variable $\overline{\rho}_i$ and design variable ρ_j, and r_{min} is the user–defined filter radius. The order of the filter is defined by the exponent q, where $q = 1$ results in the linear filter: convolution with a cone in two–dimensions. Examples of the resulting convolution functions in two–dimensions for $q = \{1, 2, 3, 4\}$ are shown in Figure 6.1. The formulation for density–based topology optimization using SIMP and including a volume filter is:

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & J\left(\boldsymbol{\rho}, \mathbf{u}\left(\boldsymbol{\rho}\right)\right) \\ \text{s.t.} \quad & \overline{\boldsymbol{\rho}} = \mathbf{H}\boldsymbol{\rho} \\ & \textstyle\sum_i^{N_e} \overline{\rho}_i v_i - \left(f\right)\left(V_0\right) \leq 0 \\ & g_i\left(\boldsymbol{\rho}, \mathbf{u}\left(\boldsymbol{\rho}\right)\right) \leq 0 \qquad i = 1 \ldots N_c \\ & 0 \leq \rho_j \leq 1 \qquad j = 1 \ldots N_e \\ & E_k\left(\overline{\rho}_k\right) = E_{min} + \overline{\rho}_k^p \left(E_0 - E_{min}\right) \qquad k = 1 \ldots N_e \\ \text{with} \quad & \mathbf{K}\left(\overline{\rho}\right)\mathbf{u} = \mathbf{f} \end{aligned} \tag{6.8}$$

Higher–order filters $q > 1$ are efficient at reducing the amount of intermediate material at the boundary due to their rapid decay (Almeida et al., 2009). The effect of filters in three–dimensional density–based topology optimization has not received as much attention as two–dimensional ones. A number of factors contribute to this: the increased complexity

of three–dimensional analysis, limited computational power, and the complexity of three–dimensional plotting. Given a filter with specific r_{min} and exponent q, the effect of the filter in two and three–dimensional space is different: the *blurring* (or *smearing*) effect is increased in three–dimensions.

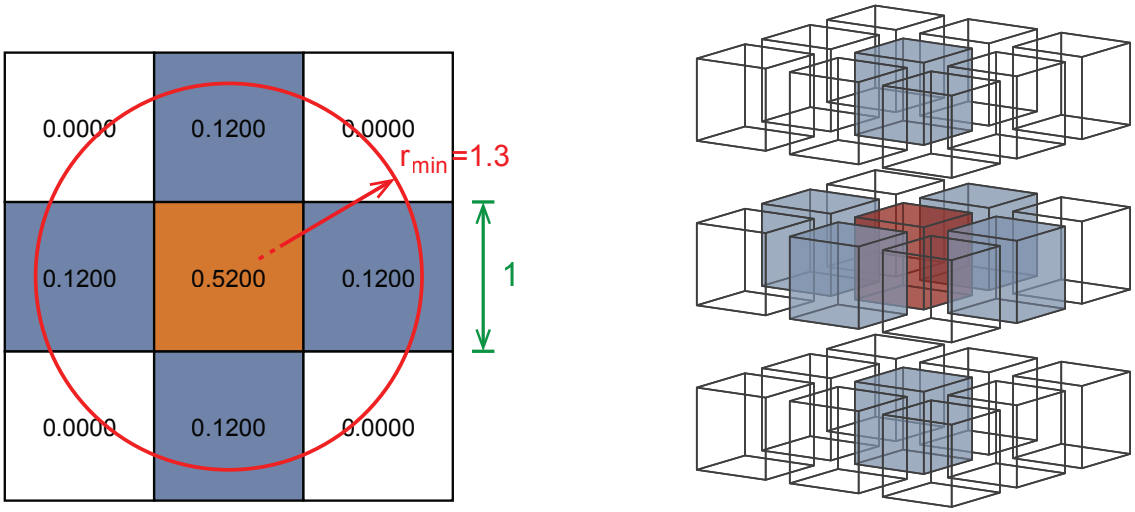


Figure 6.2: Topology optimization filters in two and three dimensions. The meshes are regular and orthogonal with elements of unit dimension. The filter is linear of size $r_{min} = 1.3$. (a) Two–dimensional filter patch: the filter weight associated with the center element is $\mathbf{H}_{ii}^{(2D)} = 0.5200$, plus 4 adjacent elements. (b) Three–dimensional filter patch: the filter weight associated with the center element is $\mathbf{H}_{ii}^{(3D)} = 0.4194$, plus 6 adjacent elements.

Consider a two–dimensional regular and orthogonal mesh with square elements of unit size, and a single design and density variable per element located at the center. Taking $r_{min} = 1.3$ and $q = 1$ in Equation (6.7), the resulting convolution kernel in two–dimensions is shown in Figure 6.2(a). Using these parameters for an equivalent three–dimensional mesh of hexahedral elements, the convolution kernel involves 6 neighboring elements instead of 4, as shown in Figure 6.2(b). The weight for the design variable at the center decreases from $\mathbf{H}_{ii}^{(2D)} = 0.5200$ to $\mathbf{H}_{ii}^{(3D)} = 0.4194$. The difference in $\mathbf{H}_{ii}$ increases with the filter radii: the elements inside a two–dimensional filter scales with r^2, compared r^3 in three–dimensions. The evolution of the weight $\mathbf{H}_{ii}$ with the filter radius for the case of regular and orthogonal

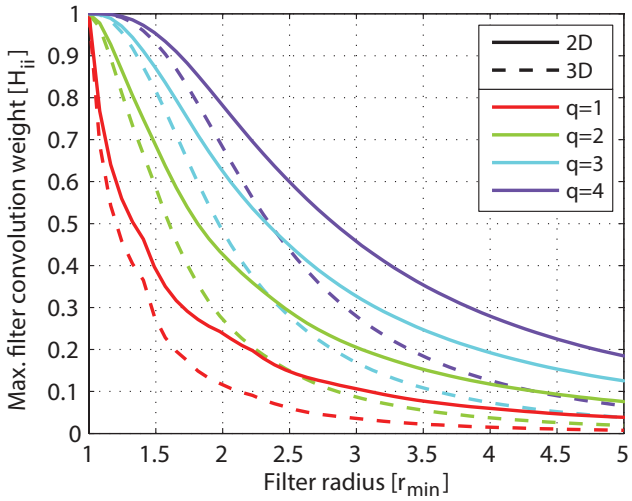


Figure 6.3: Weight coefficient for the central element $\mathbf{H}_{ii}$ for different filter radii. Curves for filters in two– and three–dimensions and for different filter order q are shown (assuming a regular and orthogonal mesh).

meshes is shown in Figure 6.3.

Consider a regular and orthogonal mesh: given a filter radius r_{min} and two–dimensional exponent $q^{(2D)}$, a three–dimensional exponent $q^{(3D)}$ can be found such that $\mathbf{H}_{ii}^{(2D)} = \mathbf{H}_{ii}^{(3D)}$. The weight coefficients of the neighboring design variables must decrease. The objective is to reduce the amount of material being *leaked* to the neighboring elements, while maintaining the filter radius unchanged (control over the minimum length–scale). Figure 6.4 plots the required $q^{(3D)}$ that results in same value for $\mathbf{H}_{ii}$.

The curves in Figure 6.4 have a discontinuity at $r_{min} = 1$ as expected, but are otherwise continuous. The curves can be considered smooth for $r_{min} \geq 1.5$. For cases with $1.5 \leq r_{min} \leq 6$ and $1 \leq q^{(2D)} \leq 3$, the following empirical expression displays good agreement with the curves in Figure 6.4:

$$q^{(3D)} = \log\left(r_{min}\right) + \frac{17}{20} q^{(2D)} + \frac{4}{57} q^{(2D)} r_{min} + \frac{4}{87} r_{min} \tag{6.9}$$

It is of interest to note that the main contributions in Equation (6.9) are: a logarithmic term for the filter radius r_{min} and a linear term for $q^{(2D)}$.

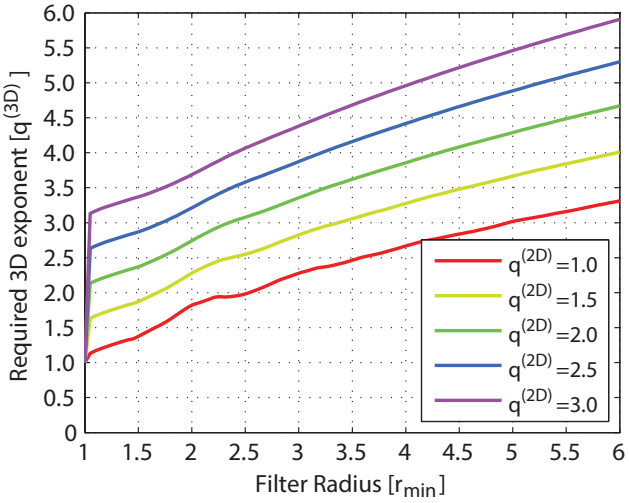


Figure 6.4: Three–dimensional filter exponent $q^{(3D)}$ required to achieve the same filter weight for the center element as in two–dimensions.

The filter exponent in three–dimensional filters $q^{(3D)}$ should be higher compared to two–dimensional filters. Maintaining the same filter radius will cause the implicit control over minimum length–scale to be similar, but not equal. These conclusions are not restricted to density filters, but extend to sensitivity filters and projection schemes (three–field SIMP) as well.

Edge–loaded cantilever example

The effect of high order filters can be further examined with an example. The three–dimensional cantilever in Figure 6.5 is clamped at one end, and loaded at the bottom edge of the opposite tip. The objective function is the minimization of structural compliance (maximization of stiffness); that is $J = \mathbf{u}^T\mathbf{K}\mathbf{u} = \mathbf{u}^T\mathbf{f}$. The problem is symmetric, and thus only a half–domain is modeled. The domain is discretized using $L_x \times L_y \times L_z = 216 \times 72 \times 72$ brick (8–node) elements ($216 \times 36 \times 72$ for the half–domain). The material's Poisson's ratio is $\nu = 0.3$ and $E_{min} = 10^{-9}E_0$. The filter radius is $r_{min} = 6$, the penalization is $p = 3$ and the volume fraction is $f = 0.1$ (10% of the domain's volume).

The cutoff (or threshold) used is $cutoff = 0.5$, i.e. the solid is defined by the domain region

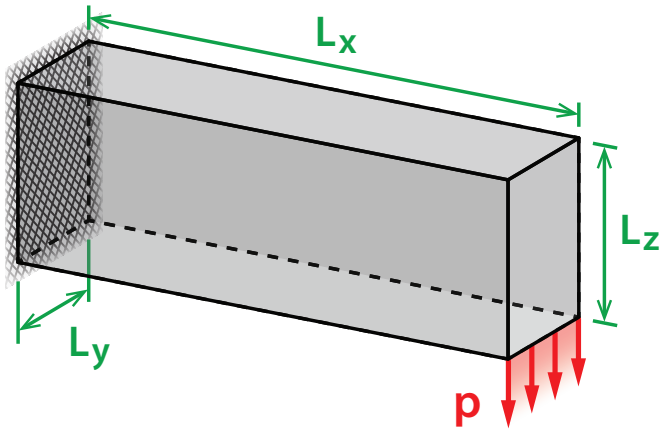


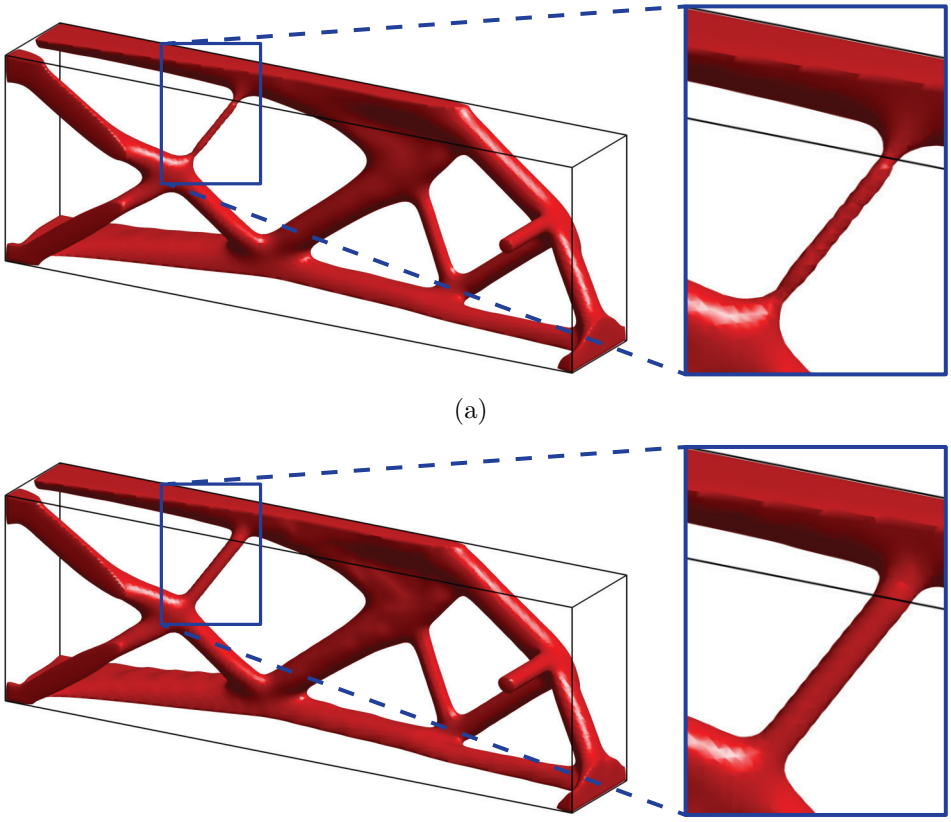
Figure 6.5: Rectangular cantilever clamped at the left side and loaded at the right by a distributed force applied at the lower edge.

with density $\rho \geq 0.5$. For the case where a linear filter is used ($q = 1$), the resulting cutoff isosurface is shown in Figure 6.6(a), including a detail on a specific member; this member suffers from an unnatural thinning at the ends. This is caused by the intermediate densities in the vicinity of the joints caused by the filter. The intermediate densities located between the members and close to the joint provide some additional stiffness, even when penalized. Consequently, the structure demands less material from the members' cross–sections to carry the load, i.e. the member cross–sectional area is reduced.

The isosurface obtained from a cubic filter ($q = 3$) (Figure 6.6(b)) does not suffer from this thinning problem. The cubic filter does not *spread* the material in the vicinity of the joint as much as the linear filter due to its rapid decay. Figures 6.7(a) and 6.7(b) slice the resulting topology along the member to highlight the intermediate densities in the vicinity of the joints.

6.1.2 Reduction of intermediate densities by continuation

Knowing that the compliance minimization problem is convex for $p = 1$ (Petersson, 1999), motivates the use of a continuation approach on the penalization parameter (Allaire and Kohn, 1993; Allaire and Francfort, 1993; Sigmund and Petersson, 1998). This consists in



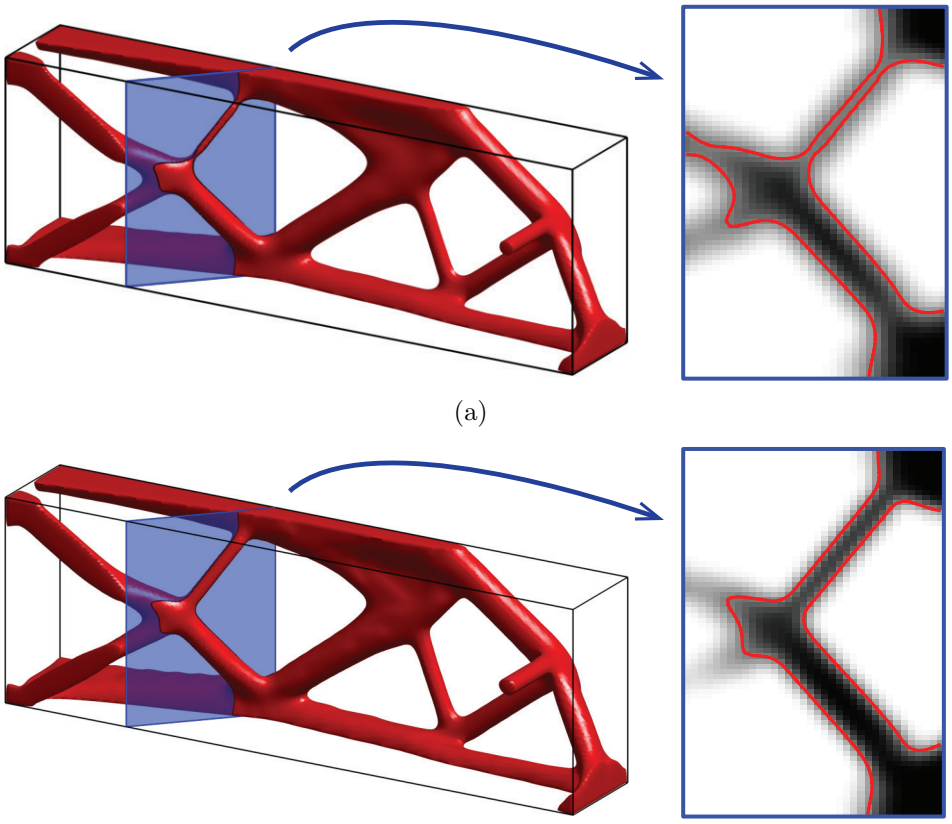
(a)

(b)

Figure 6.6: Results for the edge–loaded cantilever problem using density–based topology optimization. Plot shows the $\rho = 0.5$ isosurface (density cutoff). (a) Results using a linear filter $q = 1$, highlighting the artificial thinning of the member close to the joints. (b) Results from using a cubic filter $q = 3$ with no artificial thinning.

initially optimizing for $p = 1$, and gradually increasing the penalization value during the optimization process to reduce the amount of intermediate density values. This approach drives the solution closer towards a 0—1 design (solid—void). While this technique often converges to better designs, this cannot be guaranteed nor proven mathematically.

The original formulation (Equations (6.1) and (6.2)) was solved for a solid—void (or 0—1) solution, i.e. $\chi = \{0, 1\}$. The problem was then relaxed allowing for the continuous variable $\rho = [0, 1]$. Continuation on the penalization parameter can be interpreted as gradually re–introducing the solid—void (or 0—1) requirement. The penalization parameter can be safely increased to values $p > 4$, that in the case of constant penalization would likely converge to



(a)

(b)

Figure 6.7: Slices of the resulting density–based topology optimization with SIMP for the edge–loaded cantilever problem. (a) Isosurface and contours obtained from using a linear filter $q = 1$. (b) Isosurface and contours obtained from using a cubic filter $q = 3$.

a worse design.

Alternatively, a continuation scheme could be used on the filter radius instead, i.e. gradually decrease the filter radius during the optimization process (Sigmund and Maute, 2013). This approach may result in checkerboarded regions or small length–scale topologies if introduced too early in the optimization process, and is thus considered less robust.

Bridge loaded at the top surface

The effect of continuation at improving the resulting topologies is examined with a suitable example. The three–dimensional bridge domain in Figure 6.8 is fixed on the bottom plane

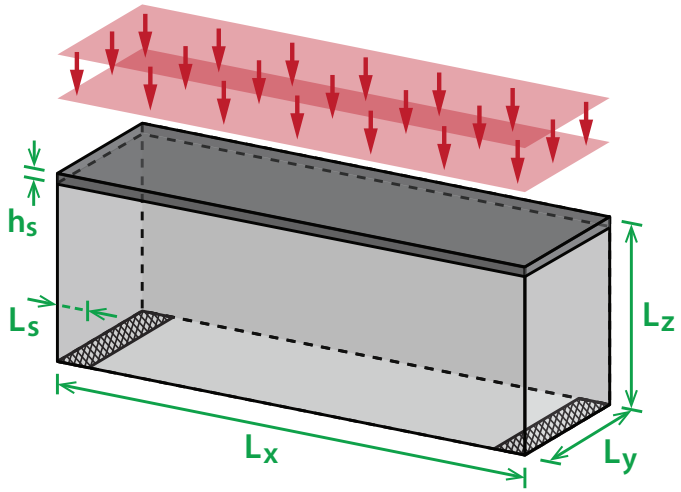


Figure 6.8: Bridge problem: Domain is loaded vertically on the top surface. The bridge slab is represented by a passive–solid region of height h_s. The domain is fixed on the bottom plane at strips of length L_s at both ends.

at strips of length $L_s = 17$ at each end (18 rows of nodes). The domain has a passive–solid slab on the top surface of height $h_s = 2$, on top of which a vertical distributed load is applied. The objective function is the minimization of structural compliance (maximization of stiffness); that is $J = \mathbf{u}^T\mathbf{K}\mathbf{u} = \mathbf{u}^T\mathbf{f}$. The problem is double–symmetric, and thus only a quarter–domain is modeled. The domain is discretized using $L_x \times L_y \times L_z = 440 \times 88 \times 88$ hexahedral (8–node) elements of unit size ($220 \times 44 \times 88$ for the quarter–domain). The material's Poisson's ratio is $\nu = 0.3$ and $E_{min} = 10^{-9}E_0$. The filter radius is $r_{min} = 5.28$, the continuation on the penalization is $p = \{1.0\,,\ 2.0\,,\ 3.0\,,\ 3.5\,,\ 4.0\,,\ 4.25\}$ and the volume fraction is $f = 0.1$ (10% of the design domain's volume).

The nature of the distributed force on the bridge causes the members to spread too thin in an effort to support the load everywhere. In doing so, the members seem to end abruptly just before the load as shown in Figure 6.9(a). This situation can be alleviated by reducing the cutoff density. This however, will result in an undesirable (artificial) increase of the volume fraction. The increased penalization in the last iterations, i.e. $p > 4$, forces the members to further define their topology. Thus preventing them from attempting to support the loaded deck everywhere as seen in Figure 6.9(b).

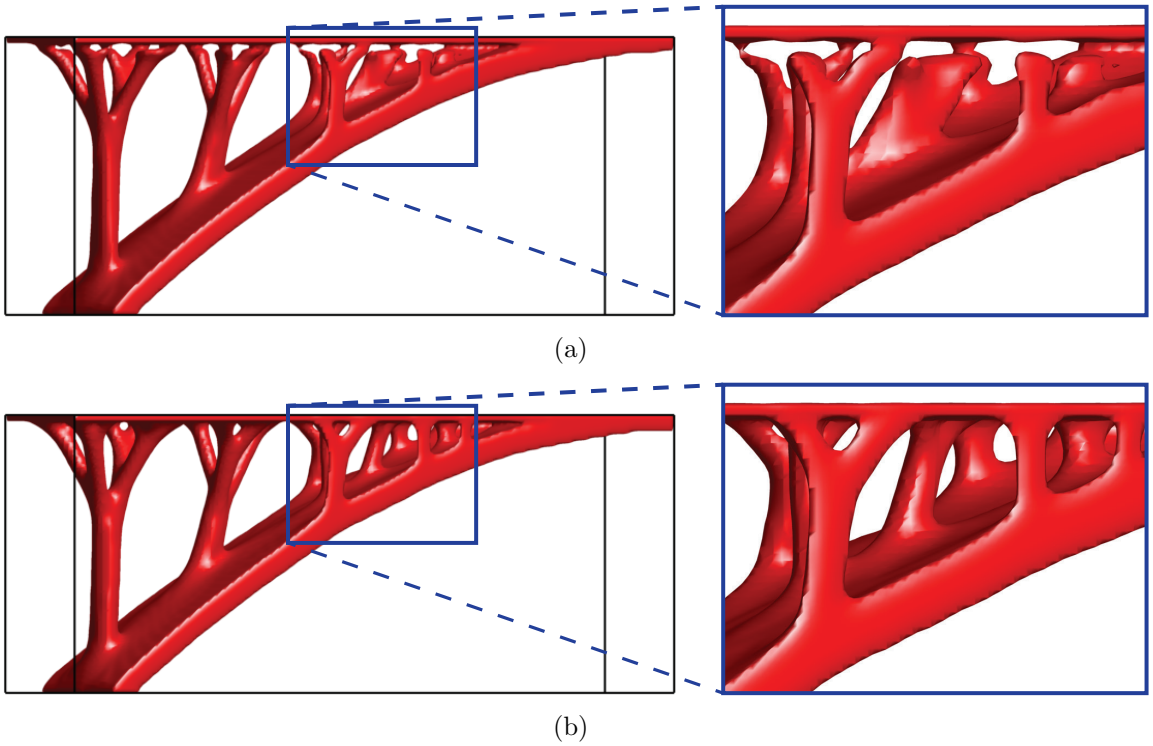


Figure 6.9: Results for the bridge problem using density–based topology optimization with SIMP. Plot shows the $\rho = 0.5$ isosurface. (a) Result using a constant penalization $p = 3$. Members end mid–air under the slab because the members spread too thin, and these intermediate densities are under the *cutoff*. (b) Results using the continuation approach for the penalization $p = \{1.0\ ,\ 2.0\ ,\ 3.0\ ,\ 3.5\ ,\ 4.0\ ,\ 4.25\}$. Members are continuous from the supports to the loaded slab.

6.2 Procedure: from the computer to your hands

Regardless of the additive manufacturing technology, a vast majority of additive manufacturing machines accept STL or stereolithography files (`*.stl`) as input (France, 2013; Lipson and Kurman, 2013). The specification for stereolithography files is relatively old and outdated. In addition, it can only describe solids by a surface tessellated into triangles.

The X3D format (`*.x3d`), itself a successor of the VRML format, is a modern royalty–free ISO standard to specify three–dimensional computer data, and is the output format choice in the present work (Brutzman and Daly, 2010). The X3D specification has implicit definitions for basic geometric primitives like the *box*, *cone*, *cylinder* and *sphere*, in addition

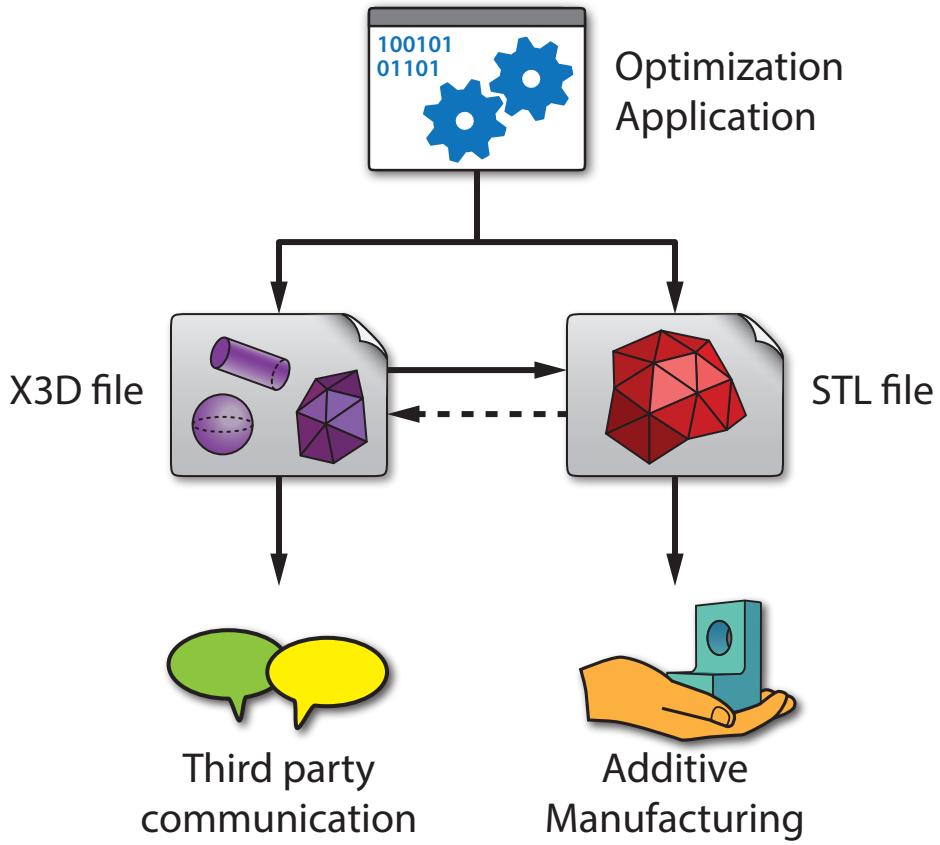


Figure 6.10: Diagram illustrating the possible file outputs (X3D and STL) and their intended purpose.

to tessellated surfaces. The number of facets or details on these implicit primitives is the responsibility of the specific renderer or interpreter. X3D has support for label definitions which can be reutilized to reduce the output size, while still being human–readable. In addition, X3D output poses additional benefits that extend beyond manufacturing: ease of communication, editing and third–party visualization.

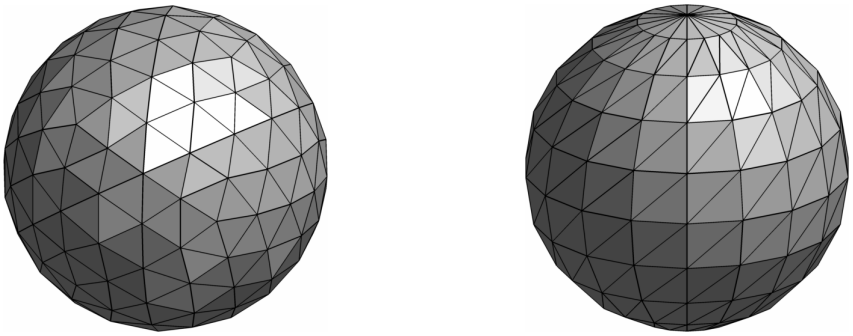
Translation from X3D to STL is straightforward and simple. In the process however, the implicit geometrical entities must be discretized. Translating from STL to X3D is also possible, however, a discretized sphere surface will not become an implicit sphere (defined by center and radius) in X3D; thus, there is a loss of information when converting from X3D to STL. For the sake of completeness and to provide options to the user, both, X3D and STL output capabilities were developed for all three types of optimal structures in this work.

The output possibilities, as well as their intended purpose are summarized in Figure 6.10.

6.2.1 Output for three–dimensional optimal ground structures

The output of three–dimensional ground structures follow a similar scheme to the one outlined in Section 5.2.4, where the members are represented by a cylinder and the joints by spheres.

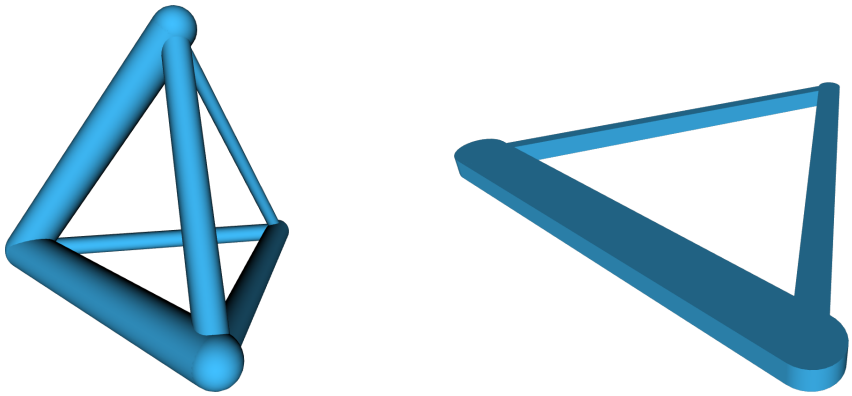
The X3D format has support for cylinders and spheres. Thus, the resulting ground structure can be exported with little to no modification.



(a) (b)

Figure 6.11: Tessellated spheres. (a) Icosphere tessellated into 324 triangles. (b) Sphere discretized using spherical coordinates using 320 triangles.

In order to directly output to an STL file, the cylinders and spheres that compose the ground structure must first be tessellated into triangles (discretized surface). This procedure causes a loss of information compared to the implicit representation of the cylinder and the sphere. The sphere is discretized with a surface partition based on an icosahedron (Figure 6.11(a)). This offers a more uniform quality of the surface discretization compared to a standard spherical coordinate discretization (Figure 6.11(b)). Figure 6.12(a) is an example of the output generated, rendered using X3DOM (Behr et al., 2009).



(a) (b)

Figure 6.12: Sample output for ground structures. (a) Three–dimensional ground structure composed of 6 members. (b) Two–dimensional ground structure composed of 3 members.

6.2.2 Output for two–dimensional optimal ground structures

The output of two–dimensional ground structures is an extrusion of the two–dimensional representation: given an extrusion height h_e, the members are slender boxes of width $w_i = a_i/h_e$. The nodes are represented by cylinders of height h_e (as opposed to spheres in the three–dimensional case).

The X3D format has support for boxes and cylinders. Thus, the resulting ground structure can be exported with little to no modification.

In order to directly output to an STL file, the boxes and cylinders that compose the ground structure must first be tessellated into triangles (discretized surface). The box's surface can be tessellated into triangles with no loss of quality. The cylinder, however, will undergo a loss of information compared to the implicit cylinder in X3D. Figure 6.12(b) is an example of the output generated, rendered using X3DOM (Behr et al., 2009).

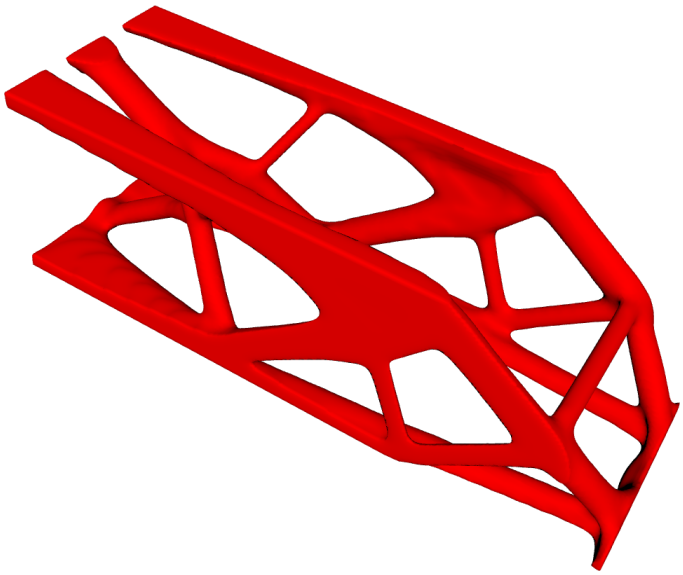


Figure 6.13: Sample rendering of an optimal edge-loaded cantilever beam optimized using SIMP. The result displayed is the $\rho = 0.5$ isosurface.

6.2.3 Output for three-dimensional density–based optimal topologies using SIMP

The density variable field at discrete spatial locations is used to obtain an isosurface (three–dimensional equivalent to a *contour line* or *isoline*). The solid is defined by this closed and water–tight isosurface. The isosurface level is typically at $\rho = 0.5$ (assuming $\rho = 0$ is void and $\rho = 1$ is solid), although values other than 0.5 may be used. The isosurface is tessellated into triangles and this data is then written to the output (STL or X3D). Translation between these formats has no information loss as data is represented by a tessellated surface in both cases. Figure 6.13 is an example of the output generated, rendered in X3DOM (Behr et al., 2009).

6.3 Rendering of optimal structures via web browser

The X3D format can be interpreted and rendered in a web browser. At the time of this writing, its usage and acceptance in three–dimensional web applications is increasing. X3DOM

(Behr et al., 2009) is a library that allows for X3D content to be interactively rendered in a browser, with requiring only JavaScript and WebGL (Khronos Group, 2014). Thus, the X3D standard has the added benefit of easily communicating three–dimensional data to third–parties without the requirement of additional software or plug–ins. In addition, the X3D model can be imported into graphical design tools used by Architects and Engineers. In the specific field of structural optimization, this can reduce the project's development time by quickly communicating design updates between involved parties. Figure 6.14 illustrates a bridge structure with terrain rendered live in a web browser.

6.4 TOPslicer — Inspector and exporter for 3D density–based topologies

Density–based topologies are represented by four–dimensional data: space coordinates $\{x, y, z\}$ and an associated density value ρ. The solid–void topology is typically obtained by calculating the isosurface at some cutoff value for ρ (post–processing). This isosurface does not provide information on the *quality of the solution*, that is, the amount of intermediate densities. To correctly visualize and examine the solution, the solid must be sliced at a plane and the resulting three–dimensional data can then be plotted and inspected.

TOPslicer is a simple graphical tool to slice three–dimensional density–based topology optimization data. In addition, the user can change the isosurface cutoff value and evaluate the change on the fly. For problems where symmetry was exploited, TOPslicer can mirror the results in one or more coordinate axis to recover the complete model. Finally, once the cutoff and symmetry conditions have been applied, and the solution is deemed acceptable from inspection, the isosurface can be exported to X3D and/or STL. Figure 6.15 shows TOPslicer working with data from the sample problem in Section 6.1.1.

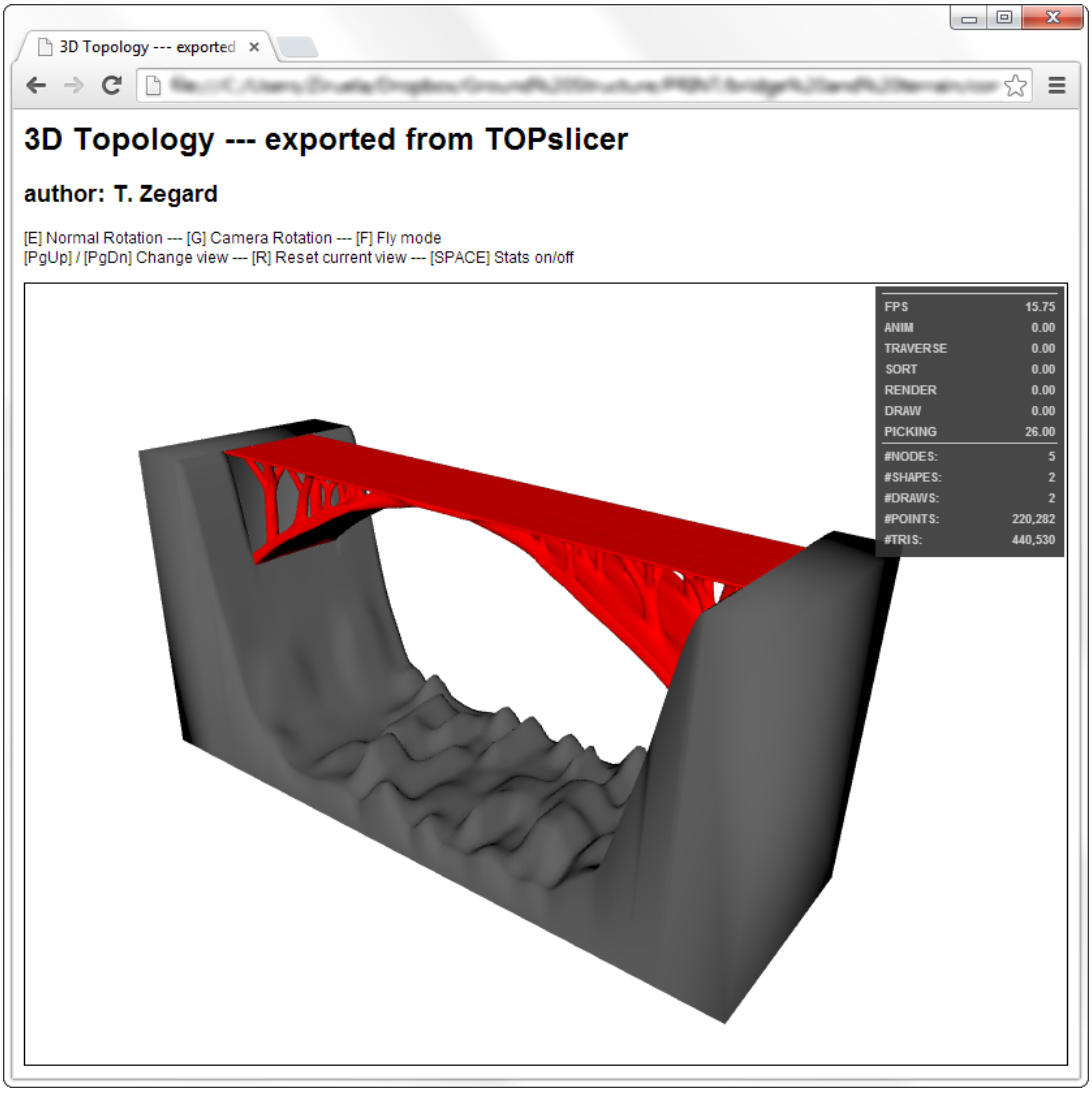


Figure 6.14: Density–based optimal structure with terrain rendered live in a web browser.

6.5 Examples of manufactured optimal structures

The optimal structures analyzed and discussed in the present work were manufactured using FDM and SLS technologies (mostly FDM). Figures 6.16 and 6.17 have pictures of manufactured optimal ground structures in two– and three–dimensional space respectively; Figure 6.18 shows manufactured optimal density–based structures; while Figure 6.19 has manufactured examples of application–focused optimal structures. Finally, the connection with the field of architecture is demonstrated by an architectural model in Figure 6.20.

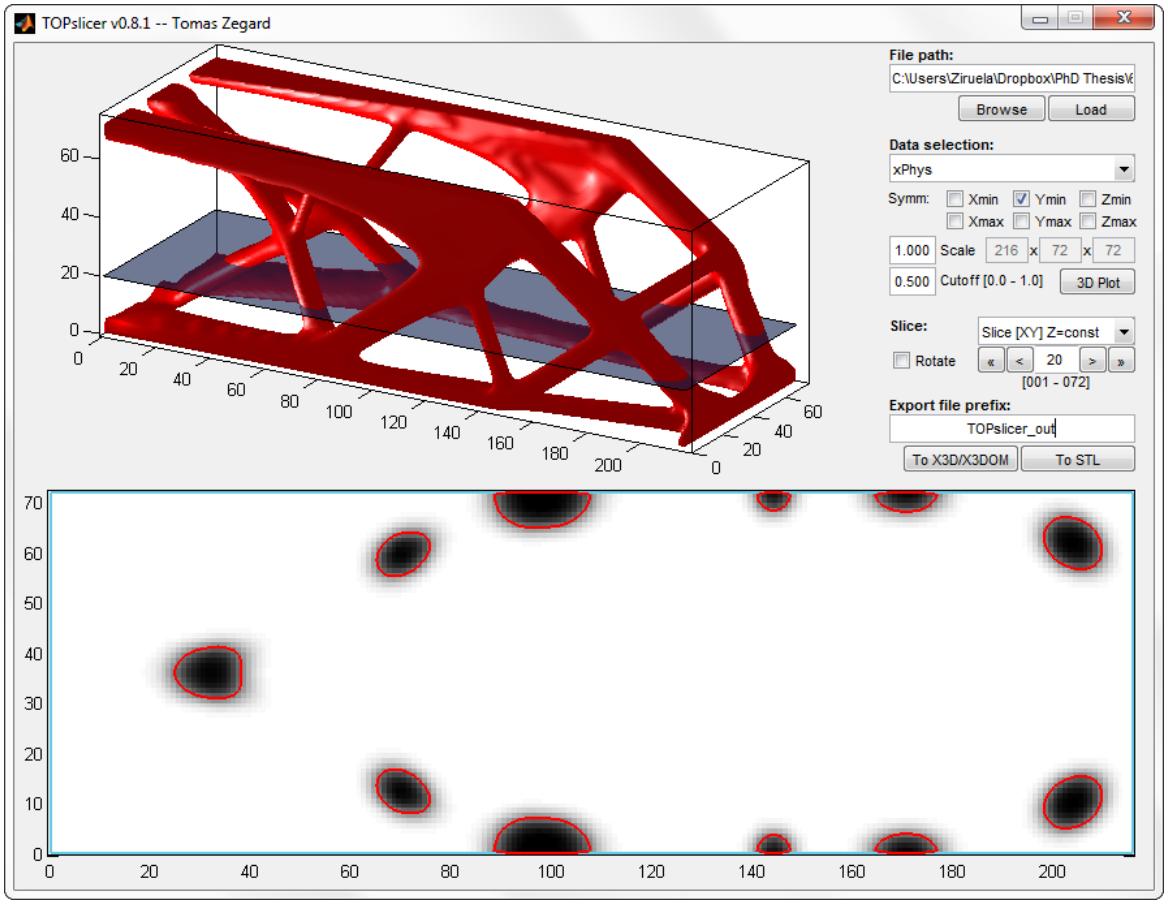
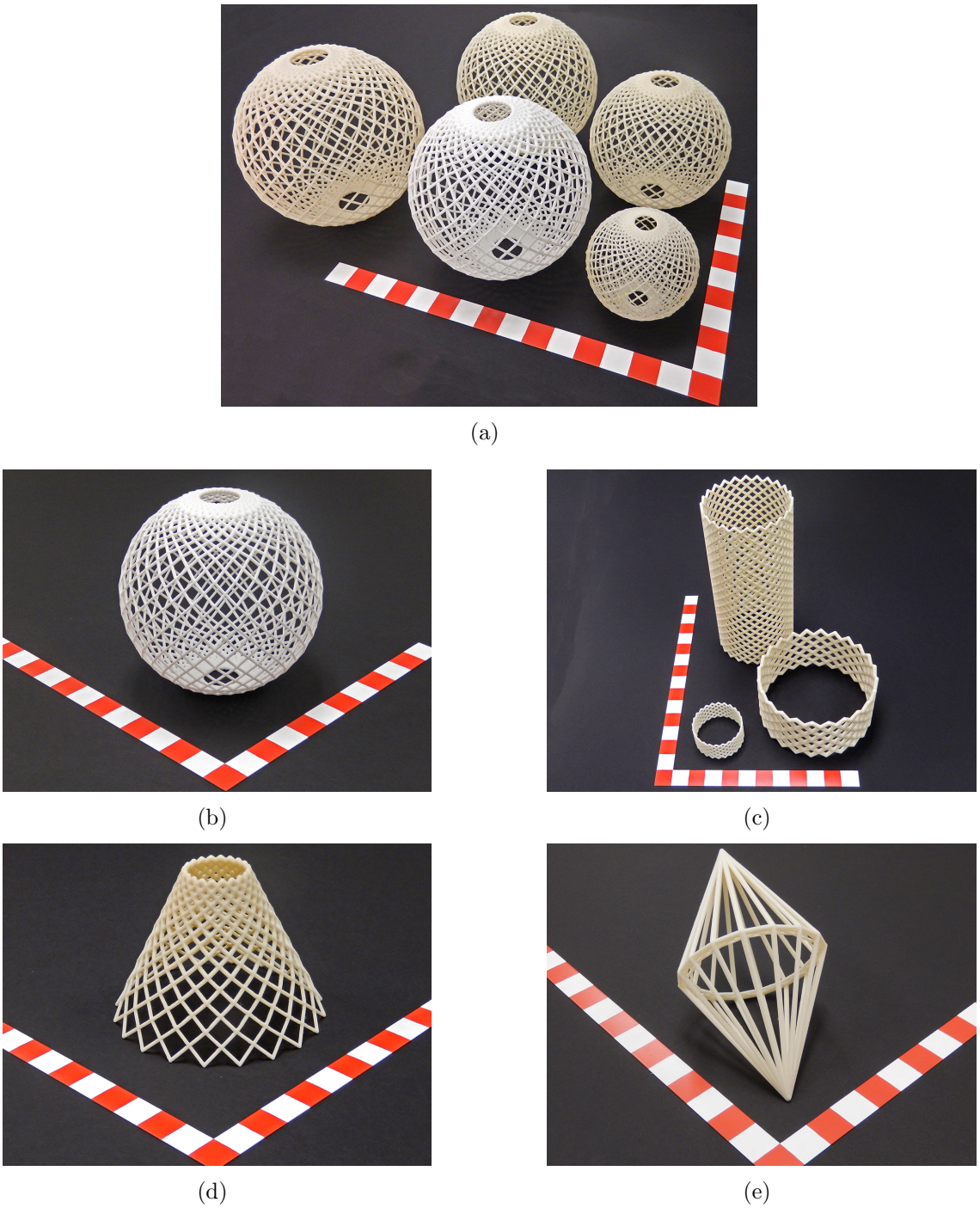


Figure 6.15: TOPslicer screenshot: The sample problem in Section 6.1.1 has symmetry applied and is being sliced to inspect the quality of the solution. The final isosurface can be directly exported for manufacture (STL) or communication and editing (X3D).

6.6 Putting it all together: workflow for an optimal human bone replacement

The human body is a highly optimized system: every component has been optimized throughout thousands of years of evolution. For this same reason, our ability to *repair* the human body after severe trauma is often partial, since it is unlikely to reach the original intended performance. Patients who suffer from traumatic bone loss, cancer, malformation or other illnesses often require bone structure replacement. These patients endure a long process of reconstruction that struggles to recreate the appearance and functionality of the



(a)

(b)

(c)

(d)

(e)

Figure 6.16: Examples of manufactured optimal three–dimensional ground structures [scales indicate inches]: (a) Torsion spheres of various sizes (Section 5.3.4). (b) Torsion sphere of 7 inches in diameter (Section 5.3.4). (c) Torsion cylinders of various sizes (Section 5.3.1). (d) Torsion cone (Section 5.3.2). (e) Diamond problem (Section 5.4.2).

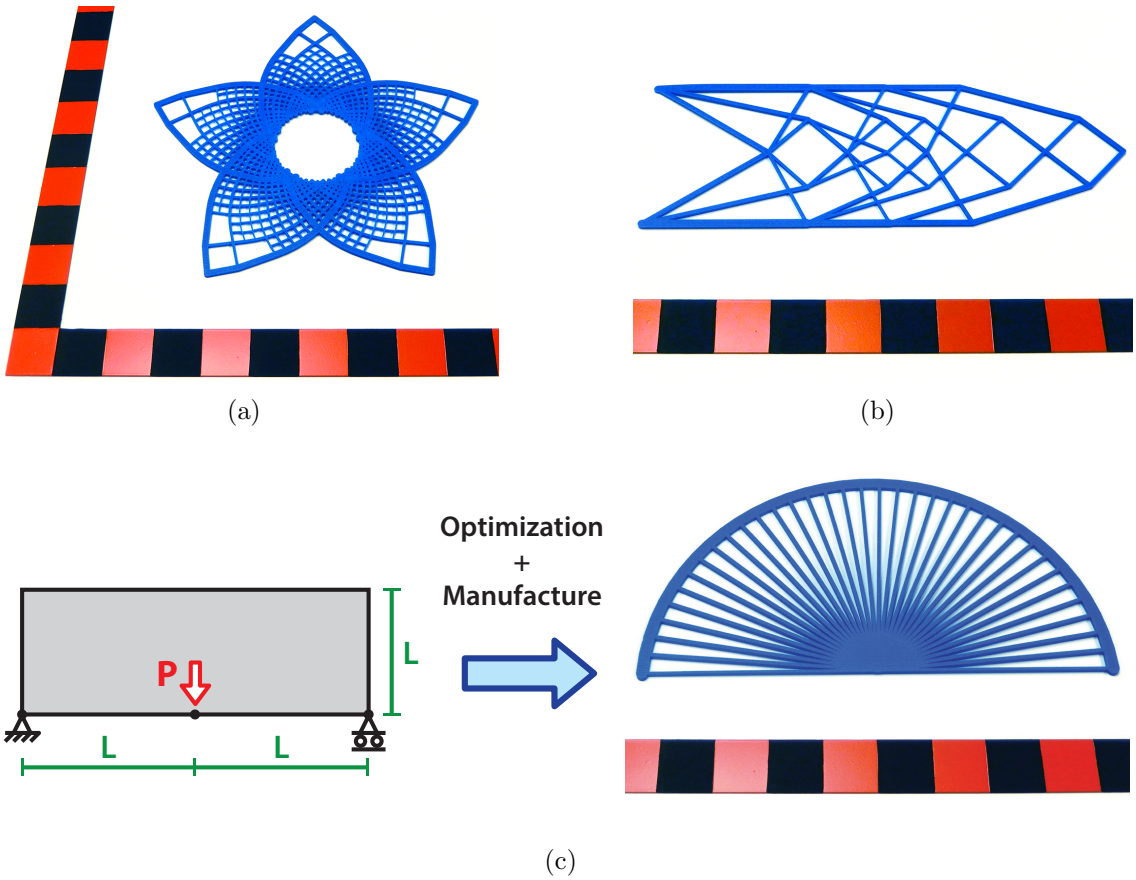
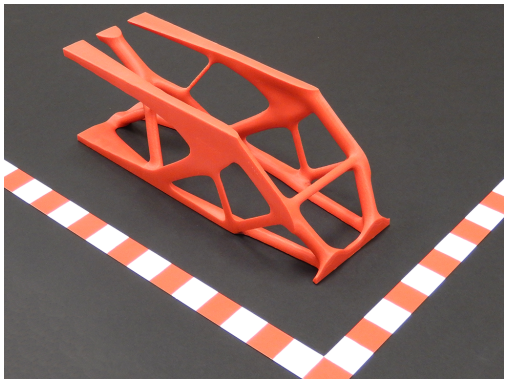


Figure 6.17: Examples of manufactured optimal two–dimensional ground structures [scales indicate inches]: (a) Flower problem (Section 4.3.1). (b) Cantilever problem (Section 4.3.1). (c) Pinwheel problem; the domain, loading and supports used to obtain the result is also provided.

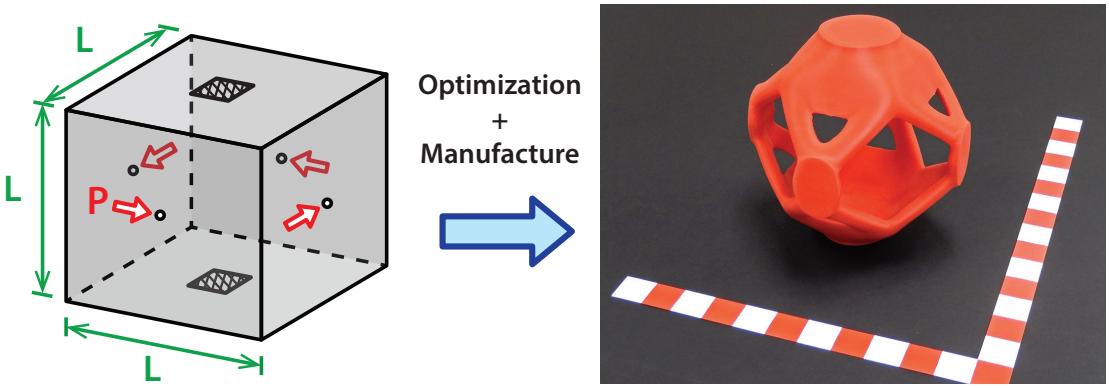
original structure.

Topology optimization and additive manufacturing technologies offer a path to potentially change the current reconstructive procedures: Given structural, biological, manufacturing and surgical requirements, usually specified by the medical doctor, an appropriate optimization problem can be formulated and solved (Sutradhar et al., 2010). The result will be a patient–specific replacement topology that is structurally optimized and functional. The solution obtained can then be manufactured, and used in the reconstructive process.

Using density–based optimization with SIMP, the process begins by defining the optimization problem: domain, loading, boundary conditions, active/passive–void zones and



(a)

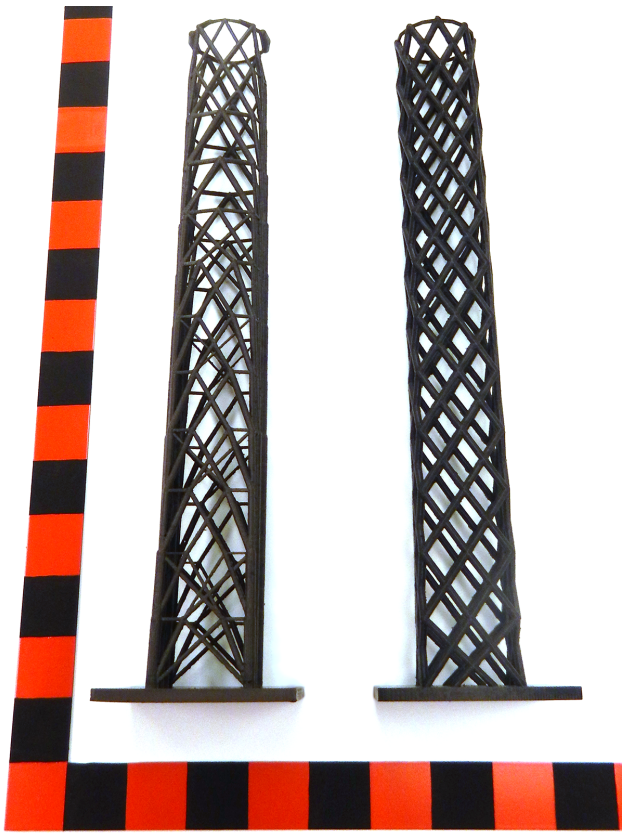


(b)

Figure 6.18: Examples of manufactured optimal three–dimensional density–based structures [scales indicate inches]: (a) Edge–loaded cantilever problem (in accordance with Figure 6.5). (b) Shear box problem; the domain, loading and supports used to generate this result are also shown [based on an example in Nguyen et al. (2009)].

optimization parameters (based on the previous work by Sutradhar et al. (2010)). For the specific problem in Figure 6.21, the domain is discretized in $108 \times 63 \times 81$ regular hexahedral elements with dimension 0.0333 *in* (dimensions are $3.6 \times 2.1 \times 2.7$ *in*), resulting in a total of $551,124$ design variables and $1,716,096$ degrees–of–freedom. The objective function is the minimization of structural compliance (maximization of stiffness); that is $J = \mathbf{u}^T\mathbf{K}\mathbf{u} = \mathbf{u}^T\mathbf{f}$. The material's Poisson's ratio is $\nu = 0.3$ and $E_{min} = 10^{-9}E_0$. The filter used has a radius of $r_{min} = 6$ *units* $= 0.2$ *in* and order $q = 3$. The SIMP penalization exponent p is increased by *continuation* from $p = 1.0$ up to $p = 4.25$.

The resulting topology is shown in Figure 6.22. The solution is well defined (Figure 6.23),



(a)

(b)

Figure 6.19: Examples of manufactured application–focused optimal structures [scales indicate inches]: (a) Laterally and torsionally loaded Lotte towers (Section 5.4.5). (b) Bridge problem (in accordance with Figure 6.8).

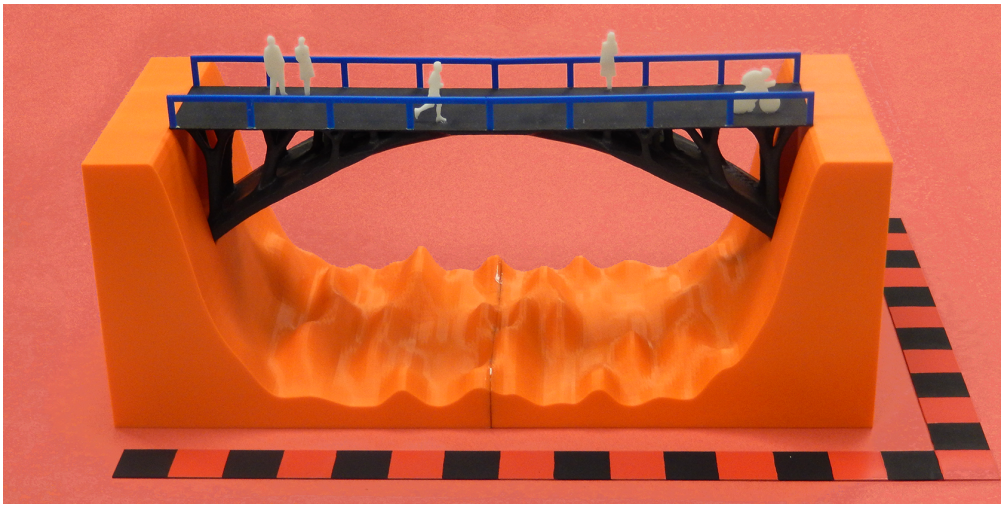


Figure 6.20: Architectural model of a topology optimized pedestrian bridge [scale indicates inches]. Model includes procedurally generated terrain, railings and people silhouettes. The contrasting colors highlight the different components in the model.

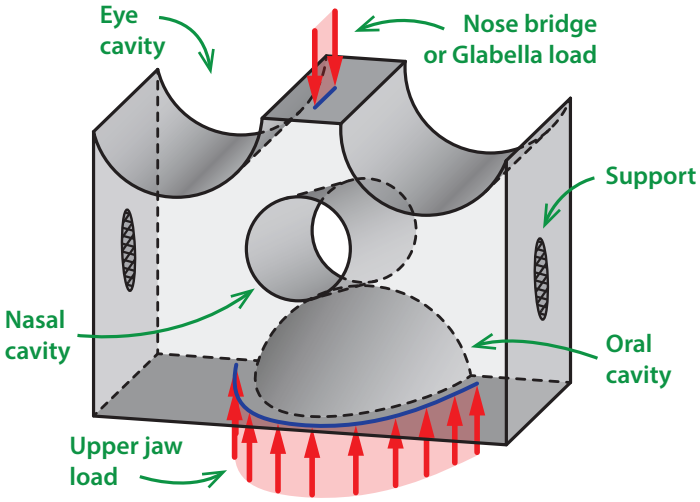


Figure 6.21: Craniofacial reconstruction problem. The design domain, loading and boundary conditions are generated procedurally, however, this particular illustration is to–scale with the results that follow.

that is, there is a relatively small amount of intermediate densities and the solution does not exhibit the issues shown in Sections 6.1.1 and 6.1.2. It should be noted that the problem is generated procedurally, and thus the domain, loading, boundary conditions and passive–void zones can be modified on a patient–specific basis. Figure 6.24 displays the resulting topology

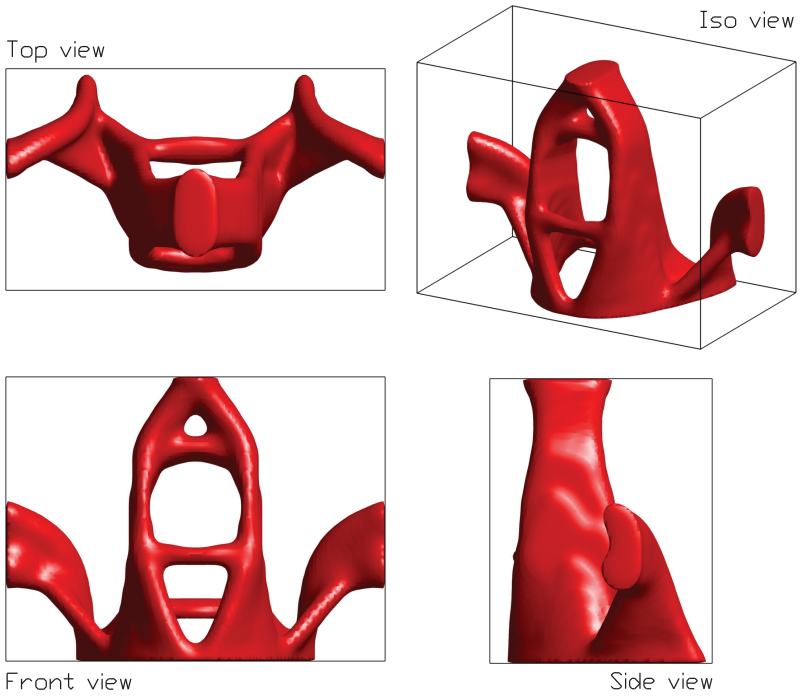


Figure 6.22: Results for the craniofacial reconstruction problem using density–based topology optimization. Plot shows the $\rho = 0.5$ isosurface (density cutoff).

in its intended position on a digital model of a skull.

The craniofacial reconstruction problem is also solved using the three–dimensional ground structure approach detailed in Chapter 5. The solution, shown in Figure 6.25, is composed of a large number of members as it is expected for this method. This result is helpful in providing information on the optimal load–paths and load transfer mechanism. The solution from this method has an overall agreement when compared to the density–based solution (Figure 6.22).

Finally, the optimized bone replacement can be manufactured and implanted in the patient. For display purposes, Figure 6.26 shows the model manufactured with FDM and attached to an upper jaw cast made from the author's teeth. However, for medical uses, the process would likely involve bio–compatible metals or human cell printing; technologies that are rapidly evolving (Salmi et al., 2012; Salmi, 2013; EOS GmbH, 2014).

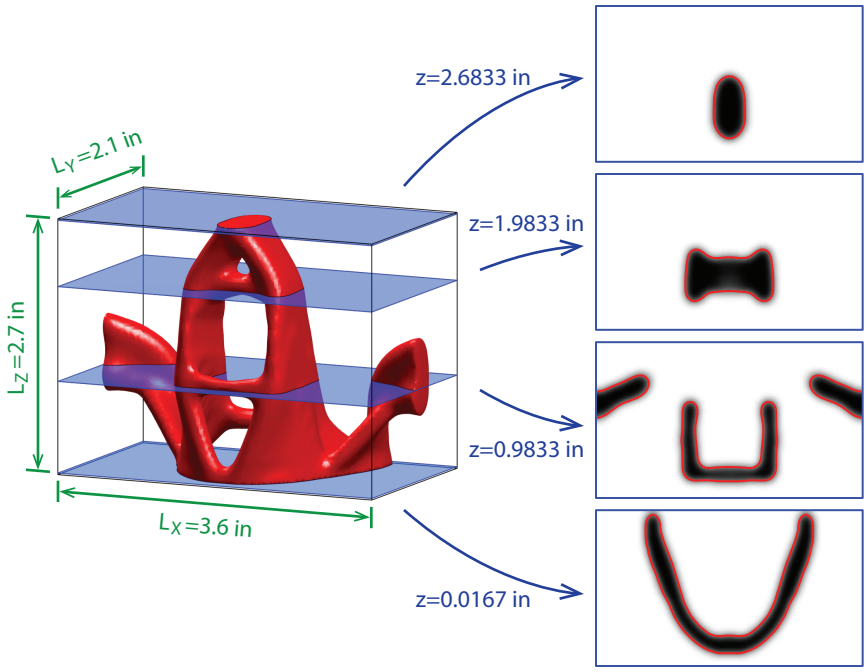


Figure 6.23: Slices of the resulting topology for the craniofacial reconstruction problem using density–based topology optimization. The slices show a well defined topology (little intermediate values), due to the continuation and higher–order filtering techniques used.

6.7 Conclusions

Additive manufacturing presents itself as the final and missing link in a complete structural optimization framework: given requirements and limitations, the problem is optimized, and finally the structure is manufactured. In addition, the three–dimensional models can be used to rapidly communicate design concepts and changes (a side effect of the proposed framework).

Two optimization methods were explored and developed: the density–based method using SIMP and the ground structure method. Both methods have strengths and weaknesses, and together provide valuable information of the optimal structural mechanism and geometry (Figure 6.27). This is specially useful at the early stages of a design; when the shape and structural mechanisms are not fully conceived.

The framework and techniques here developed are not restricted to buildings and bridges,

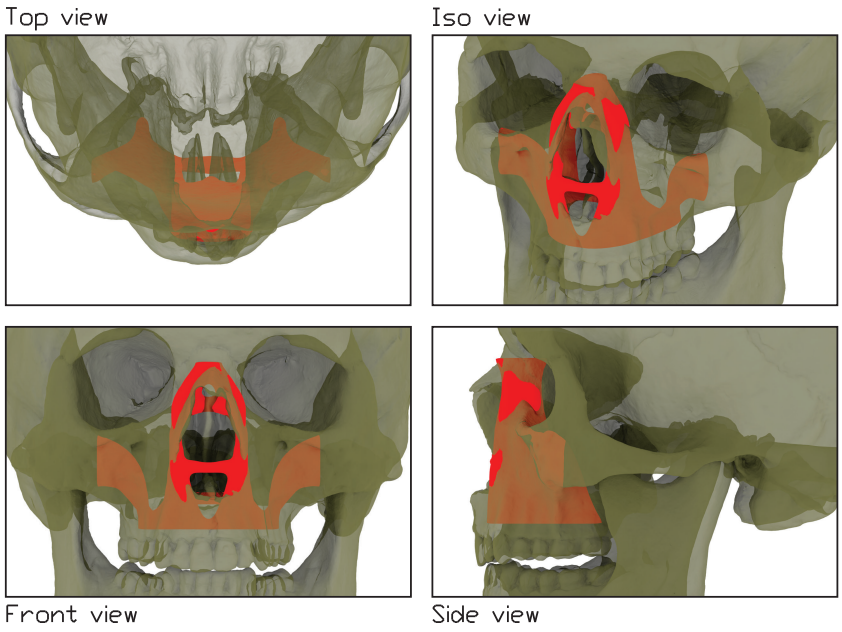


Figure 6.24: Rendering of the resulting topology for the craniofacial reconstruction problem positioned within a human skull.

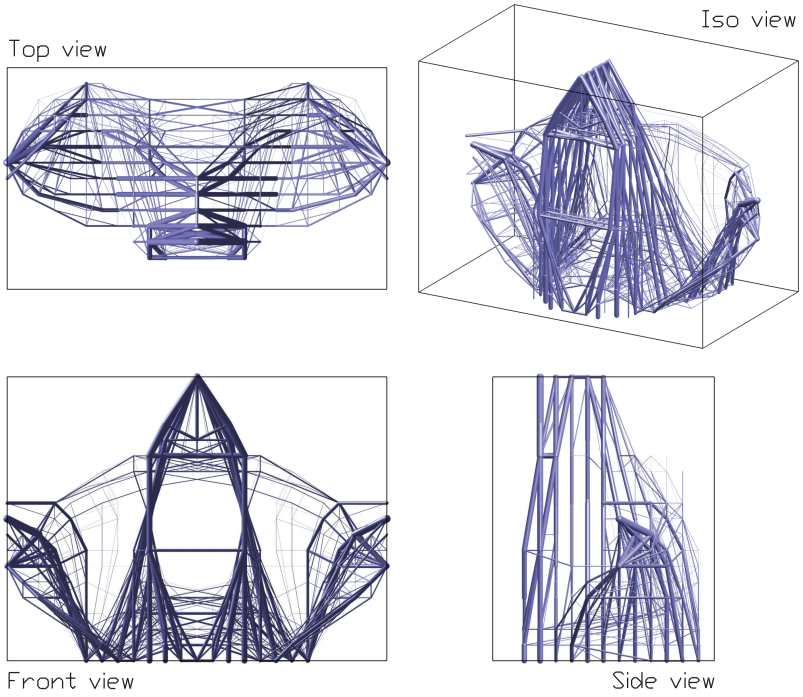
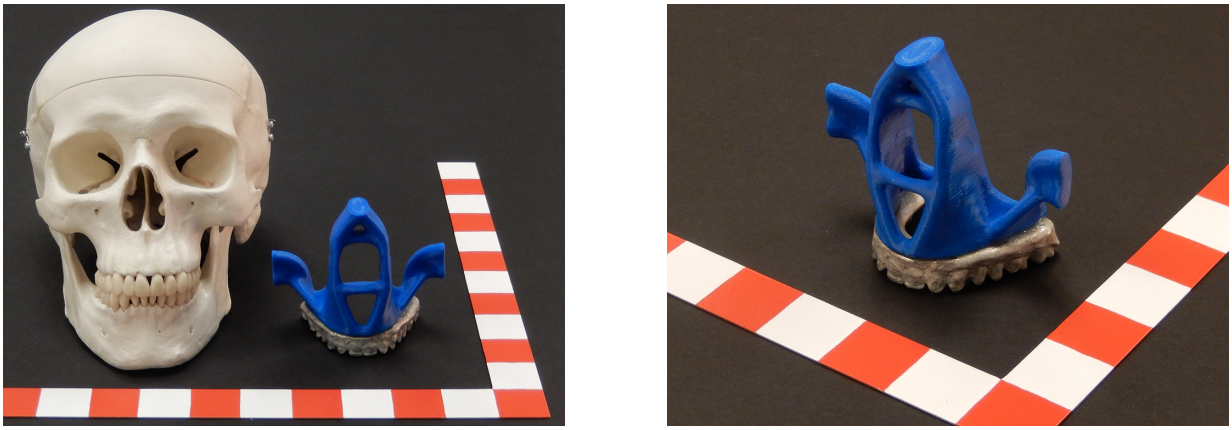


Figure 6.25: Solution for the craniofacial reconstruction problem using three–dimensional ground structures.



(a) (b)

Figure 6.26: Manufactured optimal solution for craniofacial reconstruction. Model includes an upper jaw cast in metal made from the author's teeth to serve as a reference [scales indicate inches]: (a) Frontal view with a model of a human skull for. (b) Perspective view of the model with teeth attached.

and their usage in other fields looks promising: medical applications (Sutradhar et al., 2010) and bicycle designs (Zegard and Paulino, 2013a) are few of many (yet unexplored) possibilities.

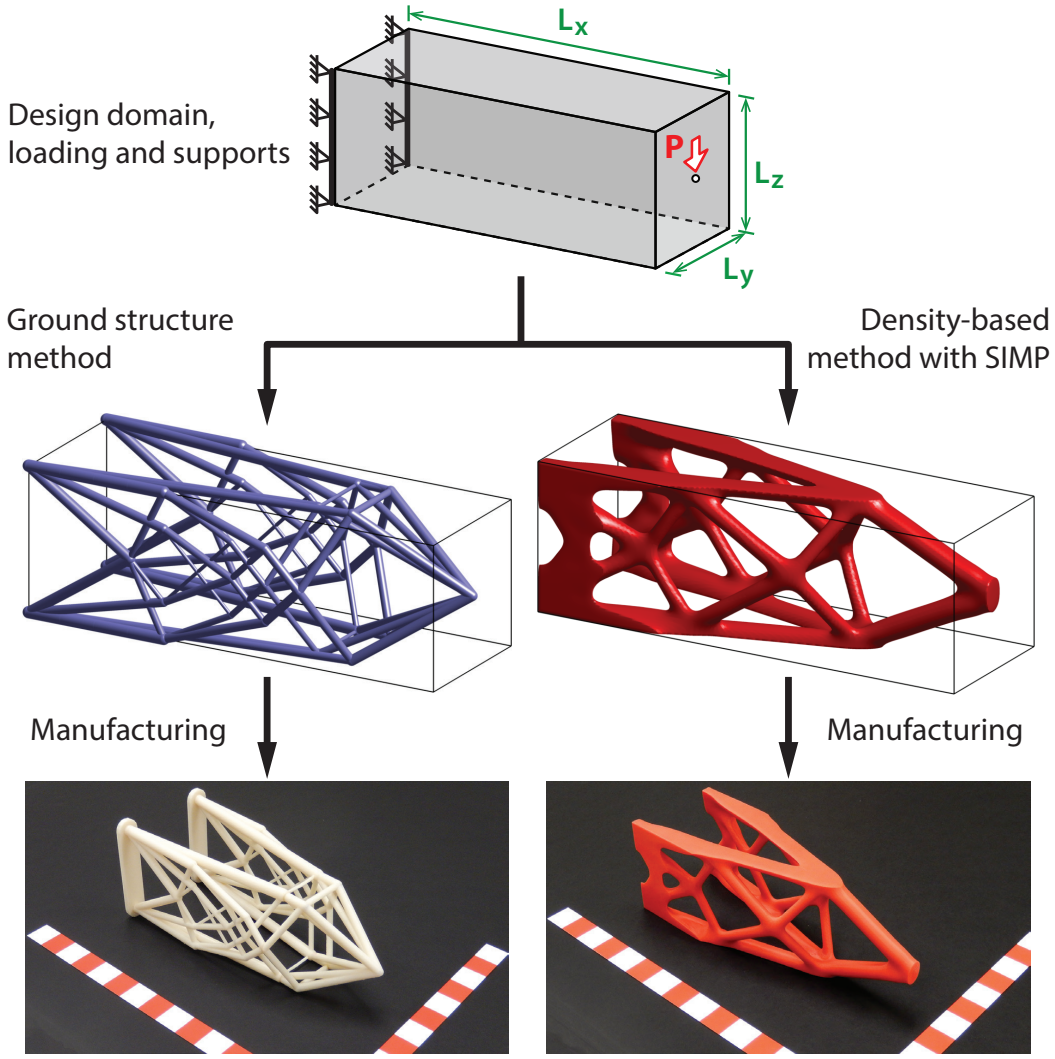


Figure 6.27: Using the framework described in this work, two distinct (but related) optimal solutions may be obtained and manufactured.

Chapter 7

Summary of conclusions and possible extensions

7.1 Summary of conclusions

The present work began with a new gradient–based formulation for structural systems combining discrete and continuum elements in Chapter 2. The goal is to optimize the layout of the discrete structural members that link to the continuum. This method is based on convolution operators to link the discrete members to the continuum. This linkage formulation introduces error into the calculations, however, the effect (or size) of the convolution operator is reduced with refinement of the continuum mesh, thus assuring convergence. This novel technique is a contribution towards the integration of discrete and continuum optimization, topics that are usually kept separate.

Optimal lateral bracing systems in two– and three–dimensional space were studied in Chapter 3. The analysis included bracing systems spanning several stories and bays, and a number of different objectives. It was shown that the optimal brace, in the sense of minimum weight and maximum stiffness, need not be the same (as is often expected). The outcome of this work is simple guidelines for the design of lateral braces. In three–dimensional space, these guidelines are different depending on wether the braces are optimized for weight/cost or for performance.

Chapter 4 described an improved algorithm for two–dimensional topology optimization using the ground structure method. The core contribution is the ground structure genera-

tion algorithm, which integrates concepts commonly used in the video–game industry—more specifically—collision tests (or hit–boxes). This approach is able to address concave geometries with the possibility of holes. The definition, as well as the collision (or restriction) zones, are built from geometrical primitives. The ground structure method, combined with this novel generation algorithm, allow the construction of complex geometries in a simple and efficient manner. Thus enabling the analysis of previously thought complicated problems. The implementation of the algorithm is freely available, and the source code for the core functions can be found in Appendix A. This source code was built with an educational focus with the hope that it proves useful to the industry and scientific communities, by providing a starting point for future developments.

The previous algorithm for two–dimensional topology optimization using the ground structure method (Chapter 4), is extended to three–dimensional space in Chapter 5. The collision tests are replaced by three–dimensional primitives used not only in the video–game industry, but on ray–tracing (computer graphics) as well. This progression is logical since real structures are three–dimensional. Analogous to the previous two–dimensional work, the implementation of the algorithm is freely available, and the source code for the core functions can be found in Appendix B.

The recent rise of additive manufacturing has made impact in multiple fields. Topology optimization can benefit from this technology for it provides means to manufacture the optimized designs—the final of a design. Chapter 6 describes a framework and best practices to manufacture optimal topologies. This work focuses on three techniques for topology optimization: three–dimensional ground structures, two–dimensional ground structures and density–based topology optimization using SIMP. In addition, the integration with more traditional structural design processes and the efficient communication between involved parties is also discussed. Examples of manufactured optimal structures are provided, some of which target fields other than civil structures in an effort to showcase the broad extent of the work.

The aggregation of the developments in this work, push the field of topology optimization closer to production environments. The manufacturing component of the work is the final stage for a complete applied optimization framework, often forgotten or disregarded. By advancing the methods and algorithms for discrete and continuum structural optimization, the integration of optimization into real projects is even more likely.

7.2 Possible extensions and future work

The work on truss layout optimization embedded in a continuum in Chapter 2, and the lateral bracing system rules in Chapter 3, are restricted to truss elements. In reality, beam elements are often used to add a degree of redundancy (safety) to the structural system. Thus, the work should consider an extension to include beam elements in the optimization.

Currently, the continuum is not optimized in the embedded truss layout optimization work. Optimizing both the discrete and the continuum is desirable, although instabilities may arise. These instabilities could potentially be addressed by optimizing the discrete and continuum in alternating cycles as opposed to together.

The ground structure implementations for concave domains in two– and three–dimensions described in Chapters 4 and 5, can become the starting points for more refined and sophisticated analysis. The adaptive ground structure method (Gilbert and Tyas, 2003) has proven to be successful at reducing the computational cost of the method, while retaining a high level of detail. In addition, local and global instabilities can also be integrated with the ground structure method (Tyas et al., 2006). The path towards integrating these techniques into production environments lies in reduced computation times and the capability to analyze large problems: parallel computing and GPU acceleration may provide the necessary tools to achieve this.

The models shown in Chapter 6 are limited to 1 foot–long. Larger structures could be split into smaller components and then assembled for larger builds. The connectivity

between the pieces requires special attention, and should be incorporated into the design process. As a proof–of–concept, a 3 foot–long long topology optimized bridge is being held by the author in Figure 7.1. This model was manufactured in 3 separate 1 foot–long pieces, and later combined into a single model. With proper considerations, this technique could be scaled up to dimensions used in civil infrastructure.

The results obtained from the optimization algorithms have to be post–processed before it can be manufactured. For the case of density–based topology optimization using SIMP, this translated in defining a cutoff density and calculate the isosurface. The structure is most likely not optimal anymore (but close). Similarly, for the case of ground structures, the member cutoff will introduce changes to the optimal topology. Models could be manufactured using high–fidelity additive manufacturing, such as Selective Laser Melting (SLM), and tested to validate (or not) the conditions of optimality.

Figure 7.1: Author holding a 3 foot–long manufactured bridge obtained with density–based topology optimization using SIMP. The bridge is made from 3 pieces of 1 foot–long pieces.

Appendix A

GRAND v1.0 source code

Table A.1 provides a brief description of the files that comprise the problem–independent modules of GRAND.

The source code for GRAND v1.0, as detailed in Table A.1, is as follows:

GRANDscript.m

```
%GRAND - Ground Structure Analysis and Design Code.
%   Tomas Zegard, Glaucio H Paulino - Version 1.0, Dec-2013

%% === MESH GENERATION LOADS/BCS ==========================================
kappa = 1.0; ColTol = 0.999999;
Cutoff = 0.002; Ng = 50; % Plot: Member Cutoff & Number of plot groups

% --- OPTION 1: POLYMESHER MESH GENERATION --------------------------------
% addpath('./PolyMesher')
% [NODE,ELEM,SUPP,LOAD] = PolyMesher(@MichellDomain,600,30);
% Lvl = 5; RestrictDomain = @RestrictMichell;
% rmpath('./PolyMesher')

% --- OPTION 2: STRUCTURED-ORTHOGONAL MESH GENERATION ---------------------
% [NODE,ELEM,SUPP,LOAD] = StructDomain(60,20,3,1,'MBB');
% Lvl = 6; RestrictDomain = []; % No restriction for box domain

% --- OPTION 3: LOAD EXTERNALLY GENERATED MESH ----------------------------
% load MeshHook
% Lvl = 10; RestrictDomain = @RestrictHook;

% load MeshSerpentine
% Lvl = 5; RestrictDomain = @RestrictSerpentine;

% load MeshMichell
% Lvl = 4; RestrictDomain = @RestrictMichell;

load MeshFlower
Lvl = 4; RestrictDomain = @RestrictFlower;

%% === GROUND STRUCTURE METHOD ============================================
PlotPolyMesh(NODE,ELEM,SUPP,LOAD) % Plot the base mesh
[BARS] = GenerateGS(NODE,ELEM,Lvl,RestrictDomain,ColTol); % Generate the GS
Nn = size(NODE,1); Ne = length(ELEM); Nb = size(BARS,1);
[BC] = GetSupports(SUPP);                % Get reaction nodes
[BT,L] = GetMatrixBT(NODE,BARS,BC,Nn,Nb); % Get equilibrium matrix
```

Table A.1: Description of function files in GRAND.

Filename	Description
`GRANDscript`	Main user script. Contains commented examples for all three available input options. Parameters are set in the file's preamble.
`GenerateGS`	Generates the ground structure for a given base mesh up to a specified level, with an optional restriction zone.
`GetMatrixBT`	Builds the nodal equilibrium matrix $\mathbf{B^T}$. The row entries corresponding to supports are removed.
`GetSupports`	Returns the global numbering of supported nodal components based on the data in `SUPP`. These are in turn used to remove these equations from the nodal equilibrium matrix $\mathbf{B^T}$ and force vector $\mathbf{f}$.
`GetVectorF`	Returns the nodal force vector based on the data in `LOAD`. The row entries corresponding to supports are removed.
`PlotBoundary`	Identifies the boundary edges and plots them.
`PlotPolyMesh`	Takes the resulting cross–sectional areas, the nodal coordinates and member definitions to plot all members with cross–sectional areas above the specified cutoff.
`rCircle`	Restriction zone primitive to test collision against a circle.
`rLine`	Restriction zone primitive to test collision against a line segment.
`rPolygon`	Restriction zone primitive to test collision against a polygon.
`rRectangle`	Restriction zone primitive to test collision against a rectangle.
`StructDomain`	Generates structured orthogonal domains and can optionally also return boundary conditions for a cantilever problem, the MBB beam and a half–space bridge.
`PolyMesher/`	Folder containing the polygonal mesher with all its related files (Talischi et al., 2012a).

```
37  [F] = GetVectorF(LOAD,BC,Nn);            % Get nodal force vector
38
39  fprintf('Mesh: Elements %d, Nodes %d, Bars %d, Level %d\n',Ne,Nn,Nb,Lvl)
40  BTBT = [BT -BT]; LL = [L; kappa*L]; sizeBTBT = whos('BTBT'); clear BT L
41  fprintf('Matrix [BT -BT]: %d x %d in %gMB (%gGB full)\n',...
42          length(F),length(LL),sizeBTBT.bytes/2^20,16*(2*Nn)*Nb/2^30)
43
44  tic, [S,vol,exitflag] = linprog(LL,[],[],BTBT,F,zeros(2*Nb,1));
45  fprintf('Objective V = %f\nlinprog CPU time = %g s\n',vol,toc);
46
47  S = reshape(S,numel(S)/2,2);  % Separate slack variables
48  A = S(:,1) + kappa*S(:,2);    % Get cross-sectional areas
49  N = S(:,1) - S(:,2);          % Get member forces
50
51  %% === PLOTTING ===========================================================
52  PlotGroundStructure(NODE,BARS,A,Cutoff,Ng)
53  PlotBoundary(ELEM,NODE)
```

`GenerateGS.m`

```
1  function [BARS]=GenerateGS(NODE,ELEM,Lvl,RestrictDomain,ColTol)
2
```

```
3  if nargin<5, ColTol=0.9999; end
4  if (nargin<4 || isempty(RestrictDomain)), RestrictDomain=@(~,~)[];
5  elseif nargin<3, error('Not enough input arguments.'), end
6
7  % Get element connectivity matrix
8  Nn = max(cellfun(@max,ELEM)); Ne = length(ELEM);
9  A1 = sparse(Nn,Nn);
10 for i=1:Ne, A1(ELEM{i},ELEM{i}) = true; end
11 A1 = A1 - speye(Nn,Nn); An = A1;
12
13 % Level 1 connectivity
14 [J,I] = find(An); % Reversed because find returns values column-major
15 BARS = [I J];
16 D = [NODE(I,1)-NODE(J,1) NODE(I,2)-NODE(J,2)];
17 L = sqrt(D(:,1).^2+D(:,2).^2);  % Length of bars
18 D = [D(:,1)./L D(:,2)./L];      % Normalized dir
19
20 % Levels 2 and above
21 for i=2:Lvl
22     Aold = An; An = logical(An*A1); Gn = An - Aold; % Get NEW bars @ level 'n'
23     [J,I] = find(Gn-diag(diag(Gn)));
24     if isempty(J), Lvl = i - 1; fprintf('-INFO- No new bars at Level %g\n',Lvl); break, end
25
26     RemoveFlag = RestrictDomain(NODE,[I J]); % Find and remove bars within restriction zone
27     I(RemoveFlag) = []; J(RemoveFlag) = [];
28
29     newD = [NODE(I,1)-NODE(J,1) NODE(I,2)-NODE(J,2)];
30     L = sqrt(newD(:,1).^2+newD(:,2).^2);
31     newD = [newD(:,1)./L newD(:,2)./L];
32
33     % Collinearity Check
34     p = 1; m = 1; RemoveFlag = zeros(size(I)); Nb = size(BARS,1);
35     for j=1:Nn
36         % Find I(p:q) - NEW bars starting @ node 'j'
37         for p=p:length(I), if I(p)>=j, break, end, end
38         for q=p:length(I), if I(q)>j, break, end, end
39         if I(q)>j, q = q - 1; end
40
41         if I(p)==j
42             % Find BARS(m:n) - OLD bars starting @ node 'j'
43             for m=1:Nb, if BARS(m,1)>=j, break, end, end
44             for n=m:Nb, if BARS(n,1)>j, break, end, end
45             if BARS(n,1)>j, n = n - 1; end
46
47             if BARS(n,1)==j
48                 % Dot products of old vs. new bars. If ~collinear: mark
49                 C = max(D(m:n,:)*newD(p:q,:)',[],1);
50                 RemoveFlag(p-1+find(C>ColTol)) = true;
51             end
52         end
53     end
54
55     % Remove collinear bars and make symmetric again. Bars that have one
56     % angle marked as collinear but the other not, will be spared
57     ind = find(RemoveFlag==false);
58     H = sparse(I(ind),J(ind),true,Nn,Nn,length(ind));
59     [J,I] = find(H+H');
60     fprintf('Lvl %2g - Collinear bars removed: %g\n',i,(length(RemoveFlag)-length(I))/2);
61
62     BARS = sortrows([BARS; I J]);
63     D = [NODE(BARS(:,1),1)-NODE(BARS(:,2),1) NODE(BARS(:,1),2)-NODE(BARS(:,2),2)];
64     L = sqrt(D(:,1).^2+D(:,2).^2);  % Length of bars
65     D = [D(:,1)./L D(:,2)./L];      % Normalized dir
66 end
67
68 % Only return bars {i,j} with i<j (no duplicate bars)
69 A = sparse(BARS(:,1),BARS(:,2),true,Nn,Nn);
```

```
[J,I] = find(tril(A)); BARS = [I J];
```

GetMatrixBT.m

```
function [BT,L]=GetMatrixBT(NODE,BARS,BC,Nn,Nb)
% Generate equilibrium matrix BT and get member lengths L
D = [NODE(BARS(:,2),1)-NODE(BARS(:,1),1) NODE(BARS(:,2),2)-NODE(BARS(:,1),2)];
L = sqrt(D(:,1).^2+D(:,2).^2);
D = [D(:,1)./L D(:,2)./L];
BT = sparse([2*BARS(:,1)-1 2*BARS(:,1) 2*BARS(:,2)-1 2*BARS(:,2)],...
            repmat((1:Nb)',1,4),[-D D],2*Nn,Nb);
BT(BC,:) = [];
```

GetSupports.m

```
function [BC]=GetSupports(SUPP)
% Return degrees-of-freedom with fixed (prescribed) displacements
Nf = sum(sum(~isnan(SUPP(:,2:3))));
BC = zeros(Nf,1); j = 0;
for i=1:size(SUPP,1)
    if ~isnan(SUPP(i,2)), j = j + 1; BC(j) = 2*SUPP(i) - 1; end
    if ~isnan(SUPP(i,3)), j = j + 1; BC(j) = 2*SUPP(i);     end
end
if j~=Nf, error('Parsing number mismatch on BCs.'), end
```

GetVectorF.m

```
function [F]=GetVectorF(LOAD,BC,Nn)
% Return nodal force vector
Nl = sum(sum(~isnan(LOAD(:,2:3))));
F = sparse([],[],[],2*Nn,1,Nl);
for i=1:size(LOAD,1)
    n = LOAD(i,1);
    if ~isnan(LOAD(i,2)), F(2*n-1) = LOAD(i,2); end
    if ~isnan(LOAD(i,3)), F(2*n) = LOAD(i,3);   end
end
F(BC) = [];
```

PlotBoundary.m

```
function []=PlotBoundary(ELEM,NODE)

% Get number of nodes, elements and edges (nodes) per element
Nn = size(NODE,1); Ne = length(ELEM); NpE = cellfun(@numel,ELEM);

FACE = sparse([],[],[],Nn,Nn,sum(NpE));
for i=1:Ne
    MyFACE = [ELEM{i}; ELEM{i}(2:end) ELEM{i}(1)];
    for j=1:NpE(i)
        if FACE(MyFACE(1,j),MyFACE(2,j))==0 % New edge - Flag it
            FACE(MyFACE(1,j),MyFACE(2,j)) = i;
            FACE(MyFACE(2,j),MyFACE(1,j)) =-i;
        elseif isnan(FACE(MyFACE(1,j),MyFACE(2,j)))
            error(sprintf('Edge [%d %d] found in >2 elements',MyFACE(:,j)))
        else % Edge belongs to 2 elements: inside domain. Lock it.
            FACE(MyFACE(1,j),MyFACE(2,j)) = NaN;
            FACE(MyFACE(2,j),MyFACE(1,j)) = NaN;
        end
    end
end
[BOUND(:,1),BOUND(:,2)] = find(FACE>0);
BOUND(:,3) = FACE(sub2ind(size(FACE),BOUND(:,1),BOUND(:,2)));
plot([NODE(BOUND(:,1),1) NODE(BOUND(:,2),1)]',[NODE(BOUND(:,1),2) NODE(BOUND(:,2),2)]','k')
```

PlotGroundStructure.m

```
function []=PlotGroundStructure(NODE,BARS,A,Cutoff,Ng)

figure('Name','GRAND v1.0 -- Zegard T, Paulino GH','NumberTitle','off')
hold on, axis equal, axis off, color=jet(Ng);

A = A/max(A); % Normalize to [0,1] areas
ind = find(A>Cutoff);
MyGroup = ceil(Ng*A(ind)); % Round up to the closest group of bars
Groups = cell(Ng,1);       % Store the indices of similar bars
for i=1:Ng, Groups{i} = ind(find(MyGroup==i)); end
for i=Ng:-1:1 % Plot each group of similar bars in a single plot call
    if ~isempty(Groups{i})
        XY = [NODE(BARS(Groups{i},1),:) NODE(BARS(Groups{i},2),:)];
        GroupArea = mean(A(Groups{i})); % Mean area for this group
        plot(XY(:,[1 3])',XY(:,[2 4])','LineWidth',5*sqrt(GroupArea),'Color',color(i,:))
    end
end
fprintf('-PLOT- Cutoff %g, Groups %g, Bars plotted %g\n',Cutoff,Ng,length(ind))
```

PlotPolyMesh.m

```
function []=PlotPolyMesh(NODE,ELEM,SUPP,LOAD)
figure, hold on, axis equal, axis off
MaxNVer = max(cellfun(@numel,ELEM));         %Max. num. of vertices in mesh
PadWNaN = @(E) [E NaN(1,MaxNVer-numel(E))];  %Pad cells with NaN
ElemMat = cellfun(PadWNaN,ELEM,'UniformOutput',false);
ElemMat = vertcat(ElemMat{:});               %Create padded element matrix
patch('Faces',ElemMat,'Vertices',NODE,'FaceColor','w');
if (nargin==4 && ~isempty(SUPP) && ~isempty(LOAD))
    plot(NODE(SUPP(:,1),1),NODE(SUPP(:,1),2),'b>','MarkerSize',8);
    plot(NODE(LOAD(:,1),1),NODE(LOAD(:,1),2),'m^','MarkerSize',8);
end
axis tight, drawnow
```

rCircle.m

```
function flag=rCircle(C,r,NODE,BARS)
% Circle with center point C and radius R
Nb = size(BARS,1);
U = NODE(BARS(:,1),:) - repmat(C,Nb,1);
V = NODE(BARS(:,2),:) - repmat(C,Nb,1);
D = V - U;
L = sqrt(D(:,1).^2 + D(:,2).^2);
D = [ D(:,1)./L D(:,2)./L ];
flag = any( [ ( sum(D.*V,2)>=0 ) .* ( sum(D.*U,2)<=0 ) .*...
              ( abs(D(:,1).*U(:,2)-D(:,2).*U(:,1))<r ) , ...
              ( U(:,1).^2+U(:,2).^2<=r^2 ) , ...
              ( V(:,1).^2+V(:,2).^2<=r^2 ) ] , 2);
```

rLine.m

```
function flag=rLine(A,B,NODE,BARS)
% Line segment between points A and B
P = NODE(BARS(:,1),:); D = NODE(BARS(:,2),:) - P; V = B - A;
C = D(:,1)*V(2) - V(1)*D(:,2);                      % cross(d,v)
Ct = (A(1)-P(:,1)).*D(:,2) - (A(2)-P(:,2)).*D(:,1); % cross(a-p,d)
Cu = (A(1)-P(:,1))*V(2) - (A(2)-P(:,2))*V(1);       % cross(a-p,v)
Ct = Ct./C; Cu = Cu./C;
% If intersection is between A-B and P-Q
flag = (Ct>0).*(Ct<1).*(Cu>0).*(Cu<1);
```

rPolygon.m

```
function flag=rPolygon(A,NODE,BARS)
% Polygon with N edges defined by A of size [N x 2]
% Get normals for each half-space (A are poly nodes in CCW)
Np= size(A,1); % Number of half-spaces
N = zeros(Np,2);
N(1:Np-1,:) = A(2:Np,:) - A(1:Np-1,:); N(Np,:) = A(1,:) - A(Np,:);
N = [ N(:,2) -N(:,1) ]; % Normal vectors for all half-spaces
% Get number of bars and initialize T
Nb= size(BARS,1);
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
Tmin = zeros(Nb,1); Tmax = ones(Nb,1);
% Loop through all halfspaces
for i=1:Np
    deno = D * N(i,:)';
    dist = ( repmat(A(i,:),Nb,1) - NODE(BARS(:,1),:) ) * N(i,:)';
    T = dist ./ deno;
    ind = find( (T>Tmin) .* (deno<0) ); Tmin(ind) = T(ind);
    ind = find( (T<Tmax) .* (deno>0) ); Tmax(ind) = T(ind);
end
% No intersection if Tmin>Tmax
flag = (Tmin<=Tmax);
```

rRectangle.m

```
function flag=rRectangle(Amin,Amax,NODE,BARS)
% Amin and Amax are the rectangle's limit coords: minimum and maximum
Nb= size(BARS,1);
Tmin = zeros(Nb,1); Tmax = ones(Nb,1);
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
for i=1:2 % Check on X (i=1) and Y (i=2)
    T1 = ( Amin(i) - NODE(BARS(:,1),i) ) ./ D(:,i);
    T2 = ( Amax(i) - NODE(BARS(:,1),i) ) ./ D(:,i);
    ind = find(T1>T2); % We require T1<T2, swap if not
    [T1(ind),T2(ind)] = deal(T2(ind),T1(ind)); % Swap operation
    Tmin = max(Tmin,T1); Tmax = min(Tmax,T2);
end
% No intersection with rectangle if Tmin>Tmax
flag = (Tmin<=Tmax);
```

StructDomain.m

```
function [NODE,ELEM,SUPP,LOAD]=StructDomain(Nx,Ny,Lx,Ly,ProblemID)
% Generate structured-orthogonal domains
[X,Y] = meshgrid(linspace(0,Lx,Nx+1),linspace(0,Ly,Ny+1));
NODE = [reshape(X,numel(X),1) reshape(Y,numel(Y),1)];
k = 0; ELEM = cell(Nx*Ny,1);
for j=1:Ny, for i=1:Nx
        k = k+1;
        n1 = (i-1)*(Ny+1)+j; n2 = i*(Ny+1)+j;
        ELEM{k} = [n1 n2 n2+1 n1+1];
end, end

if (nargin==4 || isempty(ProblemID)), ProblemID = 1; end
switch ProblemID
    case {'Cantilever','cantilever',1}
        SUPP = [(1:Ny+1)' ones(Ny+1,2)];
        LOAD = [Nx*(Ny+1)+round((Ny+1)/2) 0 -1];
    case {'MBB','Mbb','mbb',2}
        SUPP = [Nx*(Ny+1)+1 NaN 1;
                (1:Ny+1)' ones(Ny+1,1) nan(Ny+1,1)];
        LOAD = [Ny+1 0 -0.5];
    case {'Bridge','bridge',3}
```

```
        SUPP = [     1      1 1;
                Nx*(Ny+1)+1 1 1];
        LOAD = [(Ny+1)*round(Nx/2)+1 0 -1];
    otherwise
        SUPP = []; LOAD = [];
        disp('-INFO- Structured domain generated with no loads/BC')
end
```

Appendix B

GRAND3 v1.0-rc2 source code
(release candidate 2)

Table B.1 provides a brief description of the files that comprise the problem–independent modules of GRAND3.

Table B.1: Description of function files in GRAND3.

Filename	Description
`GRAND3script`	Main user script. Contains commented examples for the two available input options. Parameters are set in the file's preamble.
`GenerateGS3`	Generates the three–dimensional ground structure for a given base mesh up to a specified level, with an optional restriction zone.
`GetMatrixBT3`	Builds the nodal equilibrium matrix $\mathbf{B^T}$. The row entries corresponding to supports are removed.
`GetSupports3`	Returns the global numbering of supported nodal components based on the data in `SUPP`. These are in turn used to remove these equations from the nodal equilibrium matrix $\mathbf{B^T}$ and force vector $\mathbf{f}$.
`GetVectorF3`	Returns the nodal force vector based on the data in `LOAD`. The row entries corresponding to supports are removed.
`PlotDomain3`	Plots the domain (outer boundary only), with markers for the nodes with loads and supports.
`PlotGroundStructure3`	Takes the resulting cross–sectional areas, the nodal coordinates and member connectivity as inputs. Plots all members with cross–sectional areas above a specified cutoff.
`rBox`	Restriction zone primitive to test collision against a box with planes parallel to the coordinate axes.
`rCylinder`	Restriction zone primitive to test collision against an infinite cylinder.

Table B.1: (continued)

Filename	Description
`rDisc`	Restriction zone primitive to test collision against a flat disc.
`rQuad`	Restriction zone primitive to test collision against a flat quad.
`rRod`	Restriction zone primitive to test collision against a rod (finite cylinder).
`rSphere`	Restriction zone primitive to test collision against a sphere.
`rSurf`	Restriction zone primitive to test collision against a surface (internally calls `rTriangle` and `rQuad`).
`rTriangle`	Restriction zone primitive to test collision against a flat triangle.
`StructDomain3`	Generates structured orthogonal domains and can optionally also return boundary conditions for a three–dimensional cantilever, bridge, tripod, torsion box and pyramid problems.

The source code for GRAND3 v1.0-rc2, as detailed in Table B.1, is as follows:

`GRAND3script.m`

```
%GRAND3 - 3D Ground Structure Analysis and Design Code.
%   Tomas Zegard, Glaucio H Paulino - Version 1.0, Nov-2014

%% === MESH GENERATION LOADS/BCS ============================================
kappa = 1.0; ColTol = 0.999999;
Ff = 2; Cutoff = 0.005; % Plot: Facet factor & member Cutoff

% --- OPTION 1: STRUCTURED-ORTHOGONAL MESH GENERATION ----------------------
[NODE,ELEM,SUPP,LOAD]=StructDomain3(12,4,4,3,1,1,'Cantilever');
Lvl=3; RestrictDomain=[];

% --- OPTION 2: LOAD EXTERNALLY GENERATED MESH -----------------------------
% load MeshCylinder
% Lvl=3; RestrictDomain=[];

% load MeshTube
% Lvl=3; RestrictDomain=@RestrictTube;

% load MeshCone
% Lvl=3; RestrictDomain=[];

% load MeshBallHollow
% Lvl=3; RestrictDomain=@RestrictBallHollow;

% load MeshDiamond
% Lvl=3; RestrictDomain=@RestrictDiamond;

% load MeshCantileverWedges
% Lvl=6; RestrictDomain=[];

% load MeshCantileverHole
% Lvl=6; RestrictDomain=@RestrictCantileverHole;

% load MeshCup
% Lvl=3; RestrictDomain=@RestrictCup;

% load MeshCraneFine
% load MeshCraneCoarse
% Lvl=3; RestrictDomain=@RestrictCrane;
```

```

% load MeshLotteLat
% load MeshLotteTor
% Lvl=5; RestrictDomain=@(N,B)rSurf(RNODE,ELEM.S,N,B);

% load MeshCraniofacial
% Lvl=2; RestrictDomain=@RestrictCraniofacial;

%% === GROUND STRUCTURE METHOD ============================================
PlotDomain3(NODE,ELEM,SUPP,LOAD) % Plot the base mesh
[BARS] = GenerateGS3(NODE,ELEM,Lvl,RestrictDomain,ColTol); % Generate the GS
Nn = size(NODE,1); Nb = size(BARS,1); Ne = 0;
if isfield(ELEM,'V'), Ne = Ne + length(ELEM.V); end
if isfield(ELEM,'S'), Ne = Ne + length(ELEM.S); end
[BC] = GetSupports3(SUPP);                % Get reaction nodes
[BT,L] = GetMatrixBT3(NODE,BARS,BC,Nn,Nb); % Get equilibrium matrix
[F] = GetVectorF3(LOAD,BC,Nn);            % Get nodal force vector

fprintf('Mesh: Elements %d, Nodes %d, Bars %d, Level %d\n',Ne,Nn,Nb,Lvl)
BTBT = [BT -BT]; LL = [L; kappa*L]; sizeBTBT = whos('BTBT'); clear BT L
fprintf('Matrix [BT -BT]: %d x %d in %gMB (%gGB full)\n',...
        length(F),length(LL),sizeBTBT.bytes/2^20,16*(3*Nn)*Nb/2^30)

tic, [S,vol,exitflag,output] = linprog(LL,[],[],BTBT,F,zeros(2*Nb,1));
fprintf('Objective V = %f\nlinprog CPU time = %g s\n',vol,toc);

S = reshape(S,numel(S)/2,2);  % Separate slack variables
A = S(:,1) + kappa*S(:,2);    % Get cross-sectional areas
N = S(:,1) - S(:,2);          % Get member forces

%% === PLOTTING ===========================================================
PlotGroundStructure3(NODE,BARS,A,Cutoff,Ff)
```

GenerateGS3.m

```
function [BARS]=GenerateGS3(NODE,ELEM,Lvl,RestrictDomain,ColTol)
nargchk(nargin,3,5);
if nargin<5, ColTol=0.9999; end % Default collinear tolerance
if (nargin<4 || isempty(RestrictDomain)), RestrictDomain=@(~,~)[]; end

% Get element connectivity matrix
if isfield(ELEM,'V'), Nn(1) = max(cellfun(@max,ELEM.V)); end
if isfield(ELEM,'S'), Nn(2) = max(cellfun(@max,ELEM.S)); end
Nn = max(Nn); A1 = sparse(Nn,Nn);
if isfield(ELEM,'V'), for i=1:length(ELEM.V), A1(ELEM.V{i},ELEM.V{i}) = true; end, end
if isfield(ELEM,'S'), for i=1:length(ELEM.S), A1(ELEM.S{i},ELEM.S{i}) = true; end, end
A1 = A1 - speye(Nn,Nn); An = A1;

% Level 1 connectivity
[J,I] = find(An); % Reversed because find returns values column-major
BARS = [I J];
D = [NODE(I,1)-NODE(J,1) NODE(I,2)-NODE(J,2) NODE(I,3)-NODE(J,3)];
L = sqrt(D(:,1).^2+D(:,2).^2+D(:,3).^2);  % Length of bars
D = [D(:,1)./L D(:,2)./L D(:,3)./L];      % Normalized dir

% Levels 2 and above
for i=2:Lvl
    Aold = An; An = logical(An*A1); Gn = An - Aold; % Get NEW bars @ level 'n'
    [J,I] = find(Gn-diag(diag(Gn)));
    if isempty(J), Lvl = i - 1; fprintf('-INFO- No new bars at Level %g\n',Lvl); break, end

    RemoveFlag = RestrictDomain(NODE,[I J]); % Find and remove bars within restriction zone
    I(RemoveFlag) = []; J(RemoveFlag) = [];

    newD = [NODE(I,1)-NODE(J,1) NODE(I,2)-NODE(J,2) NODE(I,3)-NODE(J,3)];
    L = sqrt(newD(:,1).^2+newD(:,2).^2+newD(:,3).^2);
```

```
    newD = [newD(:,1)./L newD(:,2)./L newD(:,3)./L];

    % Collinearity Check
    p = 1; m = 1; RemoveFlag = zeros(size(I)); Nb = size(BARS,1);
    for j=1:Nn
        % Find I(p:q) - NEW bars starting @ node 'j'
        for p=p:length(I), if I(p)>=j, break, end, end
        for q=p:length(I), if I(q)>j, break, end, end
        if I(q)>j, q = q - 1; end

        if I(p)==j
            % Find BARS(m:n) - OLD bars starting @ node 'j'
            for m=1:Nb, if BARS(m,1)>=j, break, end, end
            for n=m:Nb, if BARS(n,1)>j, break, end, end
            if BARS(n,1)>j, n = n - 1; end

            if BARS(n,1)==j
                % Dot products of old vs. new bars. If ~collinear: mark
                C = max(D(m:n,:)*newD(p:q,:)',[],1);
                RemoveFlag(p-1+find(C>ColTol)) = true;
            end
        end
    end

    % Remove collinear bars and make symmetric again. Bars that have one
    % angle marked as collinear but the other not, will be spared
    ind = find(RemoveFlag==false);
    H = sparse(I(ind),J(ind),true,Nn,Nn,length(ind));
    [J,I] = find(H+H');
    fprintf('Lvl %2g - Collinear bars removed: %g\n',i,(length(RemoveFlag)-length(I))/2);

    BARS = sortrows([BARS; I J]);
    D = [NODE(BARS(:,1),1) - NODE(BARS(:,2),1)...
         NODE(BARS(:,1),2) - NODE(BARS(:,2),2)...
         NODE(BARS(:,1),3) - NODE(BARS(:,2),3)];
    L = sqrt(D(:,1).^2+D(:,2).^2+D(:,3).^2);  % Length of bars
    D = [D(:,1)./L D(:,2)./L D(:,3)./L];      % Normalized dir
end

% Only return bars {i,j} with i<j (no duplicate bars)
A = sparse(BARS(:,1),BARS(:,2),true,Nn,Nn);
[J,I] = find(tril(A)); BARS = [I J];
```

GetMatrixBT3.m

```
function [BT,L]=GetMatrixBT3(NODE,BARS,BC,Nn,Nb)
% Generate equilibrium matrix BT and get member lengths L
D = [NODE(BARS(:,2),1)-NODE(BARS(:,1),1)...
     NODE(BARS(:,2),2)-NODE(BARS(:,1),2)...
     NODE(BARS(:,2),3)-NODE(BARS(:,1),3)];
L = sqrt(D(:,1).^2+D(:,2).^2+D(:,3).^2);
D = [D(:,1)./L D(:,2)./L D(:,3)./L];
BT = sparse([3*BARS(:,1)-2 3*BARS(:,1)-1 3*BARS(:,1)...
             3*BARS(:,2)-2 3*BARS(:,2)-1 3*BARS(:,2)],...
             repmat((1:Nb)',1,6),[-D D],3*Nn,Nb);
BT(BC,:) = [];
```

GetSupports3.m

```
function [BC]=GetSupports3(SUPP)
% Return degrees-of-freedom with fixed (prescribed) displacements
Nf = sum(sum(~isnan(SUPP(:,2:4))));
BC = zeros(Nf,1); j = 0;
for i=1:size(SUPP,1)
```

```
    if ~isnan(SUPP(i,2)), j = j + 1; BC(j) = 3*SUPP(i) - 2; end
    if ~isnan(SUPP(i,3)), j = j + 1; BC(j) = 3*SUPP(i) - 1; end
    if ~isnan(SUPP(i,4)), j = j + 1; BC(j) = 3*SUPP(i);     end
end
if j~=Nf, error('Parsing number mismatch on BCs.'), end
```

GetVectorF3.m

```
function [F]=GetVectorF3(LOAD,BC,Nn)
% Return nodal force vector
Nl = sum(sum(~isnan(LOAD(:,2:4))));
F = sparse([],[],[],3*Nn,1,Nl);
for i=1:size(LOAD,1)
    n = LOAD(i,1);
    if ~isnan(LOAD(i,2)), F(3*n-2) = LOAD(i,2); end
    if ~isnan(LOAD(i,3)), F(3*n-1) = LOAD(i,3); end
    if ~isnan(LOAD(i,4)), F(3*n) = LOAD(i,4);   end
end
F(BC) = [];
```

PlotDomain3.m

```
function []=PlotDomain3(NODE,ELEM,SUPP,LOAD,Color)
nargchk(nargin,4,5);

Alpha = 0.15; % Alpha transparency value
if nargin<5, Color = 0.3*[1 1 1]; end % Default color is gray

%% Create index lists for every element type
ELEMlist = cell(7,1);
if isfield(ELEM,'V')
    VNp = cellfun(@numel,ELEM.V);
    VELEMtype = [VNp==8 VNp==6 VNp==5 VNp==4]; % Hexahedra, Prism & Tetrahedra
    ELEMlist{1} = find(VELEMtype(:,1)); % --- Hexahedra
    ELEMlist{2} = find(VELEMtype(:,2)); % --- Prism
    ELEMlist{3} = find(VELEMtype(:,3)); % --- Pyramid
    ELEMlist{4} = find(VELEMtype(:,4)); % --- Tetrahedra
end
if isfield(ELEM,'S')
    SNp = cellfun(@numel,ELEM.S);
    SELEMtype = [SNp==4 SNp==3 SNp==2]; % Quadrangles, Triangles & Lines
    ELEMlist{5} = find(SELEMtype(:,1)); % --- Quadrangles
    ELEMlist{6} = find(SELEMtype(:,2)); % --- Triangles
    ELEMlist{7} = find(SELEMtype(:,3)); % --- Lines
end
Ne = cellfun(@numel,ELEMlist);
FACE4 = nan(6*Ne(1)+3*Ne(2)+Ne(3)+Ne(5),4); N4 = 0; % Faces: hexa-6, prism-3, pyra-1, quad-1
FACE3 = nan(2*Ne(2)+4*Ne(3)+4*Ne(4)+Ne(6),3); N3 = 0; % Faces: prism-2, pyra-4, tetra-4, tria-1

for i=1:Ne(1) % --- Hexahedra
    FACE4(N4+(1:6),:) = ELEM.V{ELEMlist{1}(i)}([1 2 3 4; 5 6 7 8; 1 2 6 5; 4 3 7 8; 2 3 7 6; 1 4 8 5]);
    N4 = N4 + 6;
end
for i=1:Ne(2) % --- Prisms
    FACE3(N3+(1:2),:) = ELEM.V{ELEMlist{2}(i)}([1 2 3; 4 5 6]); N3 = N3 + 2;
    FACE4(N4+(1:3),:) = ELEM.V{ELEMlist{2}(i)}([1 2 5 4; 2 3 6 5; 3 1 4 6]); N4 = N4 + 3;
end
for i=1:Ne(3) % --- Pyramids
    FACE3(N3+(1:4),:) = ELEM.V{ELEMlist{3}(i)}([1 2 5; 2 3 5; 3 4 5; 4 1 5]); N3 = N3 + 4;
    FACE4(N4+1,:) = ELEM.V{ELEMlist{3}(i)}([1 2 3 4]); N4 = N4 + 1;
end
for i=1:Ne(4) % --- Tetrahedra
    FACE3(N3+(1:4),:) = ELEM.V{ELEMlist{4}(i)}([1 2 3; 1 4 2; 2 4 3; 3 4 1]); N3 = N3 + 4;
end
if Ne(5)>0, FACE4(N4+(1:Ne(5)),:) = vertcat(ELEM.S{ELEMlist{5}}); N4 = N4 + Ne(5); end
```

```
if Ne(6)>0, FACE3(N3+(1:Ne(6)),:) = vertcat(ELEM.S{ELEMlist{6}}); N3 = N3 + Ne(6); end

% --- Reorder face numbering for sorting and plot
figure, hold on, axis equal, axis off, set(gcf,'Color','w')
if ~isempty(FACE3)
    [~,ind] = min(FACE3,[],2);
    for i=1:size(FACE3,1)
        if ind(i)~=1, FACE3(i,:) = FACE3(i,[ind(i):3 1:ind(i)-1]); end
    end
    ind = find(FACE3(:,2)>FACE3(:,3)); FACE3(ind,2:3) = FACE3(ind,[3 2]);
    [FACE3,~,ind] = unique(FACE3,'rows');
    IsBoundary = sparse(ind,ones(size(ind)),ones(size(ind))); % 2 = INNER
    BOUND3 = find(IsBoundary==1);
    patch('Faces',FACE3(BOUND3,:),'Vertices',NODE,'FaceColor',Color,'FaceAlpha',Alpha);
end
if ~isempty(FACE4)
    [~,ind] = min(FACE4,[],2);
    for i=1:size(FACE4,1)
        if ind(i)~=1, FACE4(i,:) = FACE4(i,[ind(i):4 1:ind(i)-1]); end
    end
    ind = find(FACE4(:,2)>FACE4(:,end)); FACE4(ind,2:4) = fliplr(FACE4(ind,2:4));
    [FACE4,~,ind] = unique(FACE4,'rows');
    IsBoundary = sparse(ind,ones(size(ind)),ones(size(ind))); % 2 = INNER
    BOUND4 = find(IsBoundary==1);
    patch('Faces',FACE4(BOUND4,:),'Vertices',NODE,'FaceColor',Color,'FaceAlpha',Alpha);
end
if ~isempty(ELEMlist{7}) % --- Plot line elements
    LINE=cat(1,ELEM.S{ELEMlist{7}});
    plot3([NODE(LINE(:,1),1) NODE(LINE(:,2),1)]',[NODE(LINE(:,1),2) NODE(LINE(:,2),2)]',...
          [NODE(LINE(:,1),3) NODE(LINE(:,2),3)]','Color',[4 21 9]/30,'LineWidth',1.5)
end
if (nargin>2 && ~isempty(SUPP) && ~isempty(LOAD)) % --- Plot boundary conditions if available
    plot3(NODE(SUPP(:,1),1),NODE(SUPP(:,1),2),NODE(SUPP(:,1),3),'b>','MarkerSize',8);
    plot3(NODE(LOAD(:,1),1),NODE(LOAD(:,1),2),NODE(LOAD(:,1),3),'m^','MarkerSize',8);
end
view(30,20), rotate3d on, drawnow
```

PlotGroundStructure3.m

```
function []=PlotGroundStructure3(NODE,BARS,A,Cutoff,Ff)
nargchk(nargin,4,5);

RGBcolor = [0.6 0.6 1];   % Default member color
c = 1/100;                % Member size scale factor
if nargin==4, Ff = 2; end % Default facet factor (plot quality)

lim = [min(NODE); max(NODE)]; % Domain limits
dim = max(diff(lim));         % Domain dimension (for cylinder width)
A = A / max(A);               % Normalize to [0,1] areas
indC = find(A>Cutoff);        % Only plot A > cutoff * A_max
figure('Name','GRAND3 v1.0 -- Zegard T, Paulino GH','NumberTitle','off')
hold on, axis equal, axis off, set(gcf,'Color','w'), view(30,20)
axis(lim(:)' + c*dim * [-1 1 -1 1 -1 1]) % Note that max(R) = c*dim*sqrt(1)

% PLOT CYLINDERS & GET SPHERE'S RADII
nodeA = zeros(size(NODE,1),1); % Store nodal areas (sphere radii)
for i=1:length(indC)
    Nf = round(Ff*(6*sqrt(A(indC(i)))+8)); % Number of facets (empirical)
    aux = BARS(indC(i),:);
    DrawRod([NODE(aux(1),1) NODE(aux(2),1)],[NODE(aux(1),2) NODE(aux(2),2)],...
            [NODE(aux(1),3) NODE(aux(2),3)],c*dim*sqrt(A(indC(i))),RGBcolor,Nf)
    nodeA(aux) = max([nodeA(aux) A(indC(i))*[1;1]],[],2);
end
% PLOT SPHERES
indS = find(nodeA>0);
Nf = round(Ff*(4*sqrt(nodeA(indS))+6)); % Number of facets (empirical)
```

```
for i=1:max(Nf)
    [Sx,Sy,Sz] = sphere(i); Sc = repmat(reshape(RGBcolor,[1 1 3]),i+1,i+1);
    aux = find(Nf==i);
    for j=1:length(aux)
        Sr = c*dim*sqrt(nodeA(indS(aux(j))));
        surf(Sr*Sx+NODE(indS(aux(j)),1),Sr*Sy+NODE(indS(aux(j)),2),...
             Sr*Sz+NODE(indS(aux(j)),3),Sc,'EdgeColor','none')
    end
end

rotate3d on, light
fprintf('-PLOT- Cutoff %g, Facet factor %g, Bars %g, Spheres %g\n',...
        Cutoff,Ff,length(indC),length(indS))

function []=DrawRod(X,Y,Z,R,C,Nt)
% Draws rods connecting points X(:,1)-X(:,2), Y(:,1)-Y(:,2) & Z(:,1)-Z(:,2)
% Rods have radius R, color C and Nt facets.
nargchk(nargin,3,6);
Nc = size(X,1);
if nargin<6, Nt = 12; end % Draw 12 facets by default
if (nargin<5 || isempty(C)), C = 0.5*[1 1 1]; end % Default color is gray
if (nargin<4 || isempty(R)), R = ones(size(X,1),1); end % Default radius=1
if ~isequal(size(X),size(Y),size(Z)), error('Vectors must be the same lengths.'), end
if size(X,2)~=2, error('Coordinates must have size [N x 2]'), end
if isscalar(R), R = R*ones(Nc,1); end

Cc = zeros(Nt,2,3);
if size(C,1)==1, for i=1:3, Cc(:,:,i) = C(i); end, end % 3D RGB matrix

t = linspace(0,2*pi,Nt)';
qt = cos(t); rt = sin(t); % Parametrized circle for cylinder's cap
p = [diff(X,[],2) diff(Y,[],2) diff(Z,[],2)]; % Directional vector
L = sqrt(sum(p.^2,2));
p = [p(:,1)./L p(:,2)./L p(:,3)./L]; % Normalized cylinder axis director
for i=1:Nc
    if abs(p(i,3))<0.9, q = cross([0 0 1],p(i,:)); % If not pointing to +Z
    else, q = cross([0 1 0],p(i,:));
    end
    q = q / norm(q);     % Normalize second basis vector
    r = cross(q,p(i,:)); % Get the third basis vector
    r = r / norm(r);     % Normalize third basis vector

    S = R(i)*repmat(qt*q(1)+rt*r(1),1,2) + repmat(X(i,:),Nt,1);
    T = R(i)*repmat(qt*q(2)+rt*r(2),1,2) + repmat(Y(i,:),Nt,1);
    U = R(i)*repmat(qt*q(3)+rt*r(3),1,2) + repmat(Z(i,:),Nt,1);
    % If each member has different color specified
    if size(C,1)~=1, for j=1:3, Cc(:,:,j) = C(i,j); end, end
    surf(S,T,U,Cc,'EdgeColor','none')
end
```

rBox.m

```
function flag=rBox(Amin,Amax,NODE,BARS)
% Amin and Amax are the box's limit coords: minimum and maximum
Nb= size(BARS,1);
Tmin = zeros(Nb,1); Tmax = ones(Nb,1);
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
for i=1:3 % Check all 3 coordinates [X,Y,Z]
    T1 = ( Amin(i) - NODE(BARS(:,1),i) ) ./ D(:,i);
    T2 = ( Amax(i) - NODE(BARS(:,1),i) ) ./ D(:,i);
    ind = find(T1>T2); % We require T1<T2, swap if not
    [T1(ind),T2(ind)] = deal(T2(ind),T1(ind)); % Swap operation
    Tmin = max(Tmin,T1); Tmax = min(Tmax,T2);
end
% No intersection with box if Tmin>Tmax
flag = (Tmin<=Tmax);
```

rCylinder.m

```
function flag=rCylinder(A,B,r,NODE,BARS)
% Collision test for an infinite cylinder A-B, with radius r
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
V = NODE(BARS(:,1),:) - repmat(A,size(BARS,1),1);
N = B - A;

VD = sum(V.*D,2);
VN = V*N';
DN = D*N';
NN = N*N';
VV = sum(V.^2,2);
DD = sum(D.^2,2);

% Cylinder surface check
a = NN*DD - DN.^2;
b = NN*VD - DN.*VN; % Actually this is b/2
c = NN*(VV-r^2) - VN.^2;
discr = b.^2 - a.*c;
ind = find(discr>=0);
flag1 = false(size(discr,1),2);
if ~isempty(ind)
    T = [ (-b(ind)+sqrt(discr(ind)))./a(ind) (-b(ind)-sqrt(discr(ind)))./a(ind) ];
    flag1(ind,:) = (T>=0) & (T<=1);
end

% Check for point A if segment is completely contained within the cylinder
T = VN./NN;
W = V - T*N;
flag2= ( sum(W.^2,2)<=r^2 );

% If any collision, return true
flag = any([flag1 flag2],2);
```

rDisc.m

```
function flag=rDisc(A,B,r,NODE,BARS)
% Collision test for a disc centered at A, with radius r and normal towards B
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
V = NODE(BARS(:,1),:) - repmat(A,size(BARS,1),1);
N = B - A;
% Project onto disc plane
VN = V*N';
DN = D*N';
T = -VN./DN;
% Calculate distance to center
W = V + [T.*D(:,1) T.*D(:,2) T.*D(:,3)];
flag = ( sum(W.^2,2)<=r^2 ) & (T>=0) & (T<=1);
```

rQuad.m

```
function flag=rQuad(A,B1,C,B2,NODE,BARS)

Nb = size(BARS,1);

% Find intersection point
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
PA= repmat(A,Nb,1) - NODE(BARS(:,1),:);
N = cross( B1 - A , B2 - A );
T = (PA*N')./(D*N');
ind = find( (T>=0).*(T<=1) );
flag = zeros(Nb,1);

```

```
% Check along diagonal AC and identify the triangle to check
D = D(ind,:); PA= PA(ind,:);
PB1= repmat(B1,length(ind),1) - NODE(BARS(ind,1),:);
PC = repmat( C,length(ind),1) - NODE(BARS(ind,1),:);
PB2= repmat(B2,length(ind),1) - NODE(BARS(ind,1),:);
E = [D(:,2).*PC(:,3)-D(:,3).*PC(:,2) D(:,3).*PC(:,1)-D(:,1).*PC(:,3) D(:,1).*PC(:,2)-D(:,2).*PC(:,1)];
VA= sum(PB1.*E,2);
VB= sum(PB2.*E,2);
V2=-sum( PA.*E,2);
aux = [sign(VA) sign(VB) sign(V2)];
tri1 = find(abs(aux(:,1)-aux(:,3))<2);
tri2 = find(abs(aux(:,2)-aux(:,3))<2);
if ~isempty(tri1) % Indices to check for AB1C
    E(tri1,:)=[D(tri1,2).*PB1(tri1,3) - D(tri1,3).*PB1(tri1,2)...
               D(tri1,3).*PB1(tri1,1) - D(tri1,1).*PB1(tri1,3)...
               D(tri1,1).*PB1(tri1,2) - D(tri1,2).*PB1(tri1,1)]; % E1 = D x PB1
end
V31= sum(PA(tri1,:).*E(tri1,:),2);
if ~isempty(tri2) % Indices to check for BCB2
    E(tri2,:)=[D(tri2,2).*PB2(tri2,3) - D(tri2,3).*PB2(tri2,2)...
               D(tri2,3).*PB2(tri2,1) - D(tri2,1).*PB2(tri2,3)...
               D(tri2,1).*PB2(tri2,2) - D(tri2,2).*PB2(tri2,1)]; % E2 = D x PB2
end
V32= sum(PA(tri2,:).*E(tri2,:),2);

flag1 = false(length(ind),1); flag2 = false(length(ind),1);
flag1(tri1) = max(abs(aux(tri1,3)-sign(V31)),[],2)<2;
flag2(tri2) = max(abs(aux(tri2,3)-sign(V32)),[],2)<2;
flag(ind) = any([flag1 flag2],2);
```

rRod.m

```
function flag=rRod(A,B,r,NODE,BARS)
% Collision test for a rod A-B, with radius r
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
V = NODE(BARS(:,1),:) - repmat(A,size(BARS,1),1);
N = B - A;

VD = sum(V.*D,2);
VN = V*N';
DN = D*N';
NN = N*N';
VV = sum(V.^2,2);
DD = sum(D.^2,2);

% A endcap
T = -VN./DN;
W = V + [T.*D(:,1) T.*D(:,2) T.*D(:,3)];
flag1 = ( sum(W.^2,2)<=r^2 ) & (T>=0) & (T<=1);

% B endcap
T = (NN-VN)./DN;
W = V + [T.*D(:,1)-N(1) T.*D(:,2)-N(2) T.*D(:,3)-N(3)];
flag2 = ( sum(W.^2,2)<=r^2 ) & (T>=0) & (T<=1);

% Cylinder surface check
a = NN*DD - DN.^2;
b = NN*VD - DN.*VN; % Actually this is b/2
c = NN*(VV-r^2) - VN.^2;
discr = b.^2 - a.*c;
flag3 = false(size(discr,1),2);
ind = find(discr>=0);
if ~isempty(ind)
    T = [ (-b(ind)+sqrt(discr(ind)))./a(ind) (-b(ind)-sqrt(discr(ind)))./a(ind) ];
    W = V(ind,:) + [D(ind,1).*T(:,1) D(ind,2).*T(:,1) D(ind,3).*T(:,1)];
    WN= W*N';
```

```
    flag3(ind,1) = (WN>=0).*(WN<=NN).*(T(:,1)>=0).*(T(:,1)<=1);
    W = V(ind,:) + [D(ind,1).*T(:,2) D(ind,2).*T(:,2) D(ind,3).*T(:,2)];
    WN= W*N';
    flag3(ind,2) = (WN>=0) & (WN<=NN) & (T(:,2)>=0) & (T(:,2)<=1);
end

% Check for point A if segment is completely contained within the rod
T = VN./NN;
W = V - T*N;
flag4 = ( sum(W.^2,2)<=r^2 ) & (T>=0) & (T<=1);

% If any collision, return true
flag = any([flag1 flag2 flag3 flag4],2);
```

rSphere.m

```
function flag=rSphere(C,r,NODE,BARS)

D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
V = repmat(C,size(BARS,1),1) - NODE(BARS(:,1),:);

VD = sum(V.*D,2);
VV = sum(V.^2,2);
DD = sum(D.^2,2);

% Point X in L(t) is inside sphere
VDn = VD./DD;
VmW = V - [VDn.*D(:,1) VDn.*D(:,2) VDn.*D(:,3)];
flag1 = ( sum(VmW.^2,2)<=r^2 ) & (VDn>=0) & (VDn<=1);

% Point A is inside sphere
flag2 = ( VV<=r^2 );

% Point B is inside sphere
flag3 = ( VV-2*VD+DD<=r^2 );

% If any collision, return true
flag = any([flag1 flag2 flag3],2);
```

rSurf.m

```
function flag=rSurf(RNODE,RFACE,NODE,BARS)

flag = false(size(BARS,1));
for i=1:length(RFACE)
    switch length(RFACE{i})
        case {1,2}
            error('Surface facets must have 3+ nodes.')
        case 3
            flag = any([flag rTriangle(RNODE(RFACE{i}(1),:),RNODE(RFACE{i}(2),:),...
                        RNODE(RFACE{i}(3),:),NODE,BARS)],2);
        case 4
            flag = any([flag rQuad(RNODE(RFACE{i}(1),:),RNODE(RFACE{i}(2),:),...
                        RNODE(RFACE{i}(3),:),RNODE(RFACE{i}(4),:),NODE,BARS)],2);
        otherwise
            for j=3:length(RFACE{i})
                flag = any([flag rTriangle(RNODE(RFACE{i}(1),:),RNODE(RFACE{i}(j-2),:),...
                            RNODE(RFACE{i}(j),:),NODE,BARS)],2);
            end
    end
end
```

rTriangle.m

```
function flag=rTriangle(A,B,C,NODE,BARS)

Nb = size(BARS,1);

% Find intersection point
D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
PA= repmat(A,Nb,1) - NODE(BARS(:,1),:);
N = cross( B - A , C - A );
T = (PA*N')./(D*N');
ind = find( (T>=0).*(T<=1) );
flag = zeros(Nb,1);

% If R between P and Q, check if inside triangle
if ~isempty(ind)
    D = D(ind,:); PA= PA(ind,:);
    PB= repmat(B,length(ind),1) - NODE(BARS(ind,1),:);
    PC= repmat(C,length(ind),1) - NODE(BARS(ind,1),:);
    E = [D(:,2).*PC(:,3) - D(:,3).*PC(:,2)...
         D(:,3).*PC(:,1) - D(:,1).*PC(:,3)...
         D(:,1).*PC(:,2) - D(:,2).*PC(:,1)];
    V1= sum(PB.*E,2);
    V2=-sum(PA.*E,2);
    E = [D(:,2).*PB(:,3) - D(:,3).*PB(:,2)...
         D(:,3).*PB(:,1) - D(:,1).*PB(:,3)...
         D(:,1).*PB(:,2) - D(:,2).*PB(:,1)];
    V3= sum(PA.*E,2);
    aux = [sign(V1) sign(V2) sign(V3)];
    % Check for signs and consider case with zero volumes
    flag(ind) = max(aux,[],2) - min(aux,[],2)<2;
end
```

StructDomain3.m

```
function [NODE,ELEM,SUPP,LOAD]=StructDomain3(Nx,Ny,Nz,Lx,Ly,Lz,ProblemID)
nargchk(nargin,6,7);

% Generate structured-orthogonal domains
[X,Y,Z] = meshgrid(linspace(0,Lx,Nx+1),linspace(0,Ly,Ny+1),linspace(0,Lz,Nz+1));
NODE = [reshape(X,numel(X),1) reshape(Y,numel(Y),1) reshape(Z,numel(Z),1)];
Nn = (Nx+1)*(Ny+1)*(Nz+1); Ne = Nx*Ny*Nz;

ELEM.V = cell(Ne,1);
aux = [1 Ny+2 Ny+3 2 (Nx+1)*(Ny+1)+[1 Ny+2 Ny+3 2]];
for k=1:Nz,
    for j=1:Nx
        for i=1:Ny
            n = (k-1)*Ny*Nx + (j-1)*Ny + i;
            ELEM.V{n} = aux + (k-1)*(Ny+1)*(Nx+1) + (j-1)*(Ny+1) + i-1;
        end
    end
end

if (nargin<7 || isempty(ProblemID)), ProblemID = 1; end
switch ProblemID
    case {'Cantilever','cantilever',1}
        if rem(Ny,2)~=0, fprintf('INFO - Ideal Ny is EVEN.\n'), end
        SUPP = [ (1:(Nx+1)*(Ny+1):Nn)'   ones(Nz+1,3);
                 (Ny+1:(Nx+1)*(Ny+1):Nn)' ones(Nz+1,3)];
        LOAD = [round((Nz+1)/2)*(Nx+1)*(Ny+1)-round(Ny/2) 0 0 -1];
    case {'Bridge','bridge',2}
        if rem(Ny,2)~=0, fprintf('INFO - Ideal Ny is EVEN.\n'), end
        ind = find(NODE(:,1)==0);
        SUPP = [     ind      ones(size(ind)) nan(length(ind),2);
                  Nx*(Ny+1)+1        NaN           ones(1,2)     ;
                 (Nx+1)*(Ny+1)       NaN           ones(1,2)     ];
```

```
        LOAD = [Nz*(Nx+1)*(Ny+1)+round((Ny+1)/2) 0 0 -1];
    case {'Tripod','tripod',3}
        if any(rem([Nx Ny],2)), fprintf('INFO - Ideal Nx and Ny are EVEN.\n'), end
        SUPP = [      1       1 1 1;
                    Ny+1      1 1 1;
                 Nx*(Ny+1)+1  1 1 1;
                (Nx+1)*(Ny+1) 1 1 1];
        LOAD = [round(Nx/2)*(Ny+1)+round((Ny+1)/2) 0 0 -1];
    case {'Torsion','torsion',4}
        if ~all(rem([Nx Ny],2)), fprintf('INFO - Ideal Nx and Ny are ODD.\n'), end
        mid = round(Nx/2)*(Ny+1) + round(Ny/2) + [-Ny-1 -Ny 0 1]';
        SUPP = [mid zeros(4,3)];
        M = 1; % Applied moment
        f = [-Ly/Ny Lx/Nx]; dL = norm(f);
        f = (M/4)*(f/dL)*(2/dL); % Nodal loads are applied at 4 locations
        LOAD = [Nz*(Nx+1)*(Ny+1)+mid [-f 0; f(1) -f(2) 0; -f(1) f(2) 0; f 0] ];
        NODE = NODE - repmat([Lx Ly Lz]/2,Nn,1); % Center the model at the origin
    case {'Pyramid','pyramid',5}
        if any(rem([Nx Ny],2)), fprintf('INFO - Ideal Nx and Ny are EVEN.\n'), end
        SUPP = [      1        0   0  0;
                    Ny+1       0  NaN 0;
                 Nx*(Ny+1)+1  NaN NaN 0;
                (Nx+1)*(Ny+1) NaN NaN 0];
        LOAD = [Nn-round(Nx/2)*(Ny+1)-round(Ny/2) 0 0 -1];
    otherwise
        SUPP = []; LOAD = [];
        disp('-INFO- Structured domain generated with no loads/BC')
end
```

Appendix C

Collision primitive testing framework

To thoroughly test the collision primitives, simple *Graphical User Interfaces* (or GUIs) were developed in MATLAB. Segments are plotted in green by default. When a segment collides with a primitive, it changes color to red. This tool makes it simple to detect (and fix) false–positives and false–negatives. A sample GUI with tags is shown in Figure C.1, and the accompanying source code is given. Additional GUIs are shown in Figure C.2.

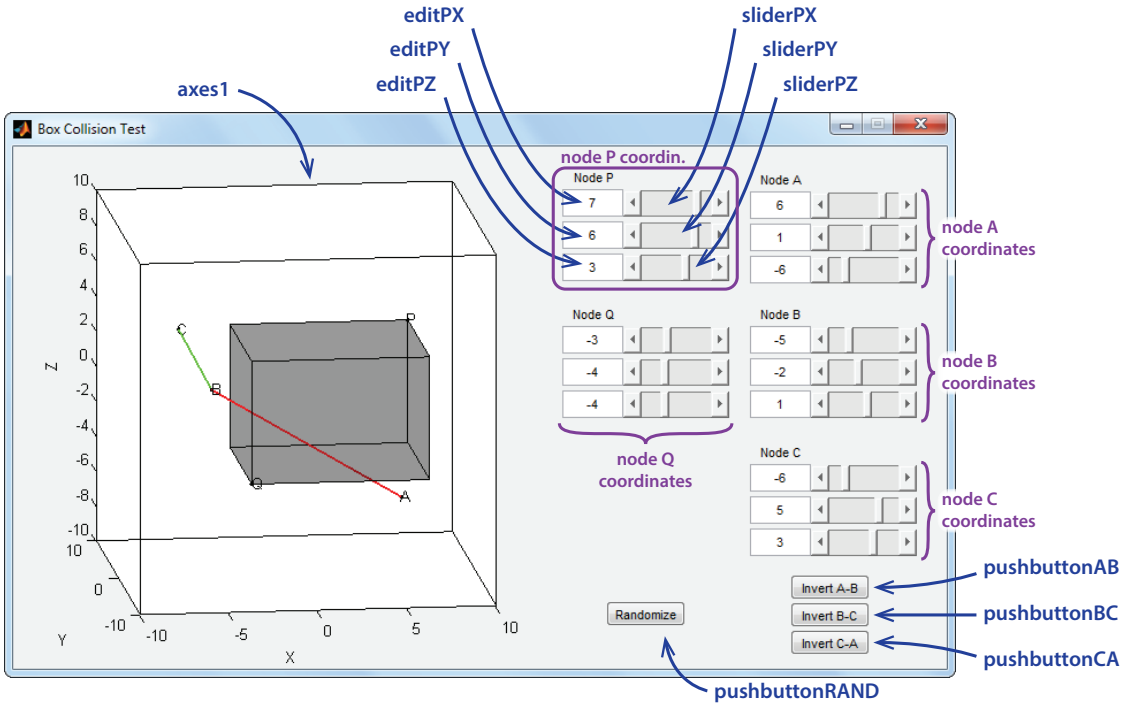


Figure C.1: Graphical user interface to test the collision of segments against a *box*. GUI object name tags are shown to match the source code callbacks.

main.m

```
function varargout = main(varargin)
% MAIN MATLAB code for main.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%      MAIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MAIN.M with the given input arguments.
%
%      MAIN('Property','Value',...) creates a new MAIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before main_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to main_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help main

% Last Modified by GUIDE v2.5 16-Jul-2014 14:07:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @main_OpeningFcn, ...
                   'gui_OutputFcn',  @main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before main is made visible.
function main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to main (see VARARGIN)

% Choose default command line output for main
handles.output = hObject;

% Default control point locations
P=[ 7  6  3];
Q=[-3 -4 -4];
A=[ 6  1 -6];
B=[-5 -2  1];
C=[-6  5  3];

set(handles.editPX ,'String',num2str(P(1)))
```

```
66  set(handles.editPY ,'String',num2str(P(2)))
67  set(handles.editPZ ,'String',num2str(P(3)))
68  set(handles.editQX ,'String',num2str(Q(1)))
69  set(handles.editQY ,'String',num2str(Q(2)))
70  set(handles.editQZ ,'String',num2str(Q(3)))
71  set(handles.editAX ,'String',num2str(A(1)))
72  set(handles.editAY ,'String',num2str(A(2)))
73  set(handles.editAZ ,'String',num2str(A(3)))
74  set(handles.editBX,'String',num2str(B(1)))
75  set(handles.editBY,'String',num2str(B(2)))
76  set(handles.editBZ,'String',num2str(B(3)))
77  set(handles.editCX,'String',num2str(C(1)))
78  set(handles.editCY,'String',num2str(C(2)))
79  set(handles.editCZ,'String',num2str(C(3)))
80  set(handles.sliderPX ,'Value',P(1))
81  set(handles.sliderPY ,'Value',P(2))
82  set(handles.sliderPZ ,'Value',P(3))
83  set(handles.sliderQX ,'Value',Q(1))
84  set(handles.sliderQY ,'Value',Q(2))
85  set(handles.sliderQZ ,'Value',Q(3))
86  set(handles.sliderAX ,'Value',A(1))
87  set(handles.sliderAY ,'Value',A(2))
88  set(handles.sliderAZ ,'Value',A(3))
89  set(handles.sliderBX,'Value',B(1))
90  set(handles.sliderBY,'Value',B(2))
91  set(handles.sliderBZ,'Value',B(3))
92  set(handles.sliderCX,'Value',C(1))
93  set(handles.sliderCY,'Value',C(2))
94  set(handles.sliderCZ,'Value',C(3))
95
96  handles.P=P; handles.Q=Q; handles.A=A; handles.B=B; handles.C=C;
97  axes(handles.axes1), view(-30,20) % Default view
98  PlotBoxAndSegment(handles)
99
100 % Update handles structure
101 guidata(hObject, handles);
102
103 % UIWAIT makes main wait for user response (see UIRESUME)
104 % uiwait(handles.figure1);
105
106
107 % --- Outputs from this function are returned to the command line.
108 function varargout = main_OutputFcn(hObject, eventdata, handles)
109 % varargout  cell array for returning output args (see VARARGOUT);
110 % hObject    handle to figure
111 % eventdata  reserved - to be defined in a future version of MATLAB
112 % handles    structure with handles and user data (see GUIDATA)
113
114 % Get default command line output from handles structure
115 varargout{1} = handles.output;
116
117
118 function editPX_Callback(hObject, eventdata, handles)
119 input=str2num(get(hObject,'String'));
120 if isempty(input)
121     set(hObject,'String',num2str(handles.P(1)))
122 else
123     handles.P(1)=input;
124     set(handles.sliderPX ,'Value',input)
125 end
126 PlotBoxAndSegment(handles)
127 guidata(hObject,handles)
128
129 function editPX_CreateFcn(hObject, eventdata, handles)
130 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
131     set(hObject,'BackgroundColor','white');
132 end
```

```
133
134
135 function editPY_Callback(hObject, eventdata, handles)
136 input=str2num(get(hObject,'String'));
137 if isempty(input)
138     set(hObject,'String',num2str(handles.P(2)))
139 else
140     handles.P(2)=input;
141     set(handles.sliderPY ,'Value',input)
142 end
143 PlotBoxAndSegment(handles)
144 guidata(hObject,handles)
145
146 function editPY_CreateFcn(hObject, eventdata, handles)
147 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
148     set(hObject,'BackgroundColor','white');
149 end
150
151
152 function editPZ_Callback(hObject, eventdata, handles)
153 input=str2num(get(hObject,'String'));
154 if isempty(input)
155     set(hObject,'String',num2str(handles.P(3)))
156 else
157     handles.P(3)=input;
158     set(handles.sliderPZ ,'Value',input)
159 end
160 PlotBoxAndSegment(handles)
161 guidata(hObject,handles)
162
163 function editPZ_CreateFcn(hObject, eventdata, handles)
164 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
165     set(hObject,'BackgroundColor','white');
166 end
167
168
169 function sliderPX_Callback(hObject, eventdata, handles)
170 input=get(hObject,'Value');
171 handles.P(1)=input;
172 set(handles.editPX ,'String',num2str(input))
173 PlotBoxAndSegment(handles)
174 guidata(hObject,handles)
175
176 function sliderPX_CreateFcn(hObject, eventdata, handles)
177 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
178     set(hObject,'BackgroundColor',[.9 .9 .9]);
179 end
180
181
182 function sliderPY_Callback(hObject, eventdata, handles)
183 input=get(hObject,'Value');
184 handles.P(2)=input;
185 set(handles.editPY ,'String',num2str(input))
186 PlotBoxAndSegment(handles)
187 guidata(hObject,handles)
188
189 function sliderPY_CreateFcn(hObject, eventdata, handles)
190 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
191     set(hObject,'BackgroundColor',[.9 .9 .9]);
192 end
193
194
195 function sliderPZ_Callback(hObject, eventdata, handles)
196 input=get(hObject,'Value');
197 handles.P(3)=input;
198 set(handles.editPZ ,'String',num2str(input))
199 PlotBoxAndSegment(handles)
```

```
guidata(hObject,handles)

function sliderPZ_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function editQX_Callback(hObject, eventdata, handles)
input=str2num(get(hObject,'String'));
if isempty(input)
    set(hObject,'String',num2str(handles.Q(1)))
else
    handles.Q(1)=input;
    set(handles.sliderQX ,'Value',input)
end
PlotBoxAndSegment(handles)
guidata(hObject,handles)

function editQX_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function editQY_Callback(hObject, eventdata, handles)
input=str2num(get(hObject,'String'));
if isempty(input)
    set(hObject,'String',num2str(handles.Q(2)))
else
    handles.Q(2)=input;
    set(handles.sliderQY ,'Value',input)
end
PlotBoxAndSegment(handles)
guidata(hObject,handles)

function editQY_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function editQZ_Callback(hObject, eventdata, handles)
input=str2num(get(hObject,'String'));
if isempty(input)
    set(hObject,'String',num2str(handles.Q(3)))
else
    handles.Q(3)=input;
    set(handles.sliderQZ ,'Value',input)
end
PlotBoxAndSegment(handles)
guidata(hObject,handles)

function editQZ_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function sliderQX_Callback(hObject, eventdata, handles)
input=get(hObject,'Value');
handles.Q(1)=input;
set(handles.editQX ,'String',num2str(input))
PlotBoxAndSegment(handles)
guidata(hObject,handles)

function sliderQX_CreateFcn(hObject, eventdata, handles)
```

```
267 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
268     set(hObject,'BackgroundColor',[.9 .9 .9]);
269 end
270
271
272 function sliderQY_Callback(hObject, eventdata, handles)
273 input=get(hObject,'Value');
274 handles.Q(2)=input;
275 set(handles.editQY ,'String',num2str(input))
276 PlotBoxAndSegment(handles)
277 guidata(hObject,handles)
278
279 function sliderQY_CreateFcn(hObject, eventdata, handles)
280 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
281     set(hObject,'BackgroundColor',[.9 .9 .9]);
282 end
283
284
285 function sliderQZ_Callback(hObject, eventdata, handles)
286 input=get(hObject,'Value');
287 handles.Q(3)=input;
288 set(handles.editQZ ,'String',num2str(input))
289 PlotBoxAndSegment(handles)
290 guidata(hObject,handles)
291
292 function sliderQZ_CreateFcn(hObject, eventdata, handles)
293 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
294     set(hObject,'BackgroundColor',[.9 .9 .9]);
295 end
296
297
298 function editAX_Callback(hObject, eventdata, handles)
299 input=str2num(get(hObject,'String'));
300 if isempty(input)
301     set(hObject,'String',num2str(handles.A(1)))
302 else
303     handles.A(1)=input;
304     set(handles.sliderAX ,'Value',input)
305 end
306 PlotBoxAndSegment(handles)
307 guidata(hObject,handles)
308
309 function editAX_CreateFcn(hObject, eventdata, handles)
310 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
311     set(hObject,'BackgroundColor','white');
312 end
313
314
315 function editAY_Callback(hObject, eventdata, handles)
316 input=str2num(get(hObject,'String'));
317 if isempty(input)
318     set(hObject,'String',num2str(handles.A(2)))
319 else
320     handles.A(2)=input;
321     set(handles.sliderAY ,'Value',input)
322 end
323 PlotBoxAndSegment(handles)
324 guidata(hObject,handles)
325
326 function editAY_CreateFcn(hObject, eventdata, handles)
327 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
328     set(hObject,'BackgroundColor','white');
329 end
330
331
332 function editAZ_Callback(hObject, eventdata, handles)
333 input=str2num(get(hObject,'String'));
```

```
334 if isempty(input)
335     set(hObject,'String',num2str(handles.A(3)))
336 else
337     handles.A(3)=input;
338     set(handles.sliderAZ ,'Value',input)
339 end
340 PlotBoxAndSegment(handles)
341 guidata(hObject,handles)
342
343 function editAZ_CreateFcn(hObject, eventdata, handles)
344 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
345     set(hObject,'BackgroundColor','white');
346 end
347
348
349 function sliderAX_Callback(hObject, eventdata, handles)
350 input=get(hObject,'Value');
351 handles.A(1)=input;
352 set(handles.editAX ,'String',num2str(input))
353 PlotBoxAndSegment(handles)
354 guidata(hObject,handles)
355
356 function sliderAX_CreateFcn(hObject, eventdata, handles)
357 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
358     set(hObject,'BackgroundColor',[.9 .9 .9]);
359 end
360
361
362 function sliderAY_Callback(hObject, eventdata, handles)
363 input=get(hObject,'Value');
364 handles.A(2)=input;
365 set(handles.editAY ,'String',num2str(input))
366 PlotBoxAndSegment(handles)
367 guidata(hObject,handles)
368
369 function sliderAY_CreateFcn(hObject, eventdata, handles)
370 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
371     set(hObject,'BackgroundColor',[.9 .9 .9]);
372 end
373
374
375 function sliderAZ_Callback(hObject, eventdata, handles)
376 input=get(hObject,'Value');
377 handles.A(3)=input;
378 set(handles.editAZ ,'String',num2str(input))
379 PlotBoxAndSegment(handles)
380 guidata(hObject,handles)
381
382 function sliderAZ_CreateFcn(hObject, eventdata, handles)
383 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
384     set(hObject,'BackgroundColor',[.9 .9 .9]);
385 end
386
387
388 function editBX_Callback(hObject, eventdata, handles)
389 input=str2num(get(hObject,'String'));
390 if isempty(input)
391     set(hObject,'String',num2str(handles.B(1)))
392 else
393     handles.B(1)=input;
394     set(handles.sliderBX,'Value',input)
395 end
396 PlotBoxAndSegment(handles)
397 guidata(hObject,handles)
398
399 function editBX_CreateFcn(hObject, eventdata, handles)
400 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
401     set(hObject,'BackgroundColor','white');
402 end
403
404
405 function editBY_Callback(hObject, eventdata, handles)
406 input=str2num(get(hObject,'String'));
407 if isempty(input)
408     set(hObject,'String',num2str(handles.B(2)))
409 else
410     handles.B(2)=input;
411     set(handles.sliderBY,'Value',input)
412 end
413 PlotBoxAndSegment(handles)
414 guidata(hObject,handles)
415
416 function editBY_CreateFcn(hObject, eventdata, handles)
417 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
418     set(hObject,'BackgroundColor','white');
419 end
420
421
422 function editBZ_Callback(hObject, eventdata, handles)
423 input=str2num(get(hObject,'String'));
424 if isempty(input)
425     set(hObject,'String',num2str(handles.B(3)))
426 else
427     handles.B(3)=input;
428     set(handles.sliderBZ,'Value',input)
429 end
430 PlotBoxAndSegment(handles)
431 guidata(hObject,handles)
432
433 function editBZ_CreateFcn(hObject, eventdata, handles)
434 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
435     set(hObject,'BackgroundColor','white');
436 end
437
438
439 function sliderBX_Callback(hObject, eventdata, handles)
440 input=get(hObject,'Value');
441 handles.B(1)=input;
442 set(handles.editBX,'String',num2str(input))
443 PlotBoxAndSegment(handles)
444 guidata(hObject,handles)
445
446 function sliderBX_CreateFcn(hObject, eventdata, handles)
447 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
448     set(hObject,'BackgroundColor',[.9 .9 .9]);
449 end
450
451
452 function sliderBY_Callback(hObject, eventdata, handles)
453 input=get(hObject,'Value');
454 handles.B(2)=input;
455 set(handles.editBY,'String',num2str(input))
456 PlotBoxAndSegment(handles)
457 guidata(hObject,handles)
458
459 function sliderBY_CreateFcn(hObject, eventdata, handles)
460 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
461     set(hObject,'BackgroundColor',[.9 .9 .9]);
462 end
463
464
465 function sliderBZ_Callback(hObject, eventdata, handles)
466 input=get(hObject,'Value');
467 handles.B(3)=input;
```

```
468 set(handles.editBZ,'String',num2str(input))
469 PlotBoxAndSegment(handles)
470 guidata(hObject,handles)
471
472 function sliderBZ_CreateFcn(hObject, eventdata, handles)
473 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
474     set(hObject,'BackgroundColor',[.9 .9 .9]);
475 end
476
477
478 function editCX_Callback(hObject, eventdata, handles)
479 input=str2num(get(hObject,'String'));
480 if (isempty(input) || input<=0)
481     set(hObject,'String',num2str(handles.C(1)))
482 else
483     handles.C(1)=input;
484     set(handles.sliderCX,'Value',input)
485 end
486 PlotBoxAndSegment(handles)
487 guidata(hObject,handles)
488
489 function editCX_CreateFcn(hObject, eventdata, handles)
490 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
491     set(hObject,'BackgroundColor','white');
492 end
493
494
495 function editCY_Callback(hObject, eventdata, handles)
496 input=str2num(get(hObject,'String'));
497 if (isempty(input) || input<=0)
498     set(hObject,'String',num2str(handles.C(2)))
499 else
500     handles.C(2)=input;
501     set(handles.sliderCY,'Value',input)
502 end
503 PlotBoxAndSegment(handles)
504 guidata(hObject,handles)
505
506 function editCY_CreateFcn(hObject, eventdata, handles)
507 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
508     set(hObject,'BackgroundColor','white');
509 end
510
511
512 function editCZ_Callback(hObject, eventdata, handles)
513 input=str2num(get(hObject,'String'));
514 if (isempty(input) || input<=0)
515     set(hObject,'String',num2str(handles.C(3)))
516 else
517     handles.C(3)=input;
518     set(handles.sliderCZ,'Value',input)
519 end
520 PlotBoxAndSegment(handles)
521 guidata(hObject,handles)
522
523 function editCZ_CreateFcn(hObject, eventdata, handles)
524 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
525     set(hObject,'BackgroundColor','white');
526 end
527
528
529 function sliderCX_Callback(hObject, eventdata, handles)
530 input=get(hObject,'Value');
531 handles.C(1)=input;
532 set(handles.editCX,'String',num2str(input))
533 PlotBoxAndSegment(handles)
534 guidata(hObject,handles)
```

```
535
536 function sliderCX_CreateFcn(hObject, eventdata, handles)
537 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
538     set(hObject,'BackgroundColor',[.9 .9 .9]);
539 end
540
541
542 function sliderCY_Callback(hObject, eventdata, handles)
543 input=get(hObject,'Value');
544 handles.C(2)=input;
545 set(handles.editCY,'String',num2str(input))
546 PlotBoxAndSegment(handles)
547 guidata(hObject,handles)
548
549 function sliderCY_CreateFcn(hObject, eventdata, handles)
550 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
551     set(hObject,'BackgroundColor',[.9 .9 .9]);
552 end
553
554
555 function sliderCZ_Callback(hObject, eventdata, handles)
556 input=get(hObject,'Value');
557 handles.C(3)=input;
558 set(handles.editCZ,'String',num2str(input))
559 PlotBoxAndSegment(handles)
560 guidata(hObject,handles)
561
562 function sliderCZ_CreateFcn(hObject, eventdata, handles)
563 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
564     set(hObject,'BackgroundColor',[.9 .9 .9]);
565 end
566
567
568 function pushbuttonRAND_Callback(hObject, eventdata, handles)
569 P=round(rand(1,3)*200-100)/10;
570 Q=round(rand(1,3)*200-100)/10;
571 while any(Q<P), Q=round(rand(1,3)*200-100)/10; end
572 A=round(rand(1,3)*200-100)/10;
573 B=round(rand(1,3)*200-100)/10;
574 C=round(rand(1,3)*200-100)/10;
575 set(handles.editPX ,'String',num2str(P(1)))
576 set(handles.editPY ,'String',num2str(P(2)))
577 set(handles.editPZ ,'String',num2str(P(3)))
578 set(handles.editQX ,'String',num2str(Q(1)))
579 set(handles.editQY ,'String',num2str(Q(2)))
580 set(handles.editQZ ,'String',num2str(Q(3)))
581 set(handles.editAX ,'String',num2str(A(1)))
582 set(handles.editAY ,'String',num2str(A(2)))
583 set(handles.editAZ ,'String',num2str(A(3)))
584 set(handles.editBX,'String',num2str(B(1)))
585 set(handles.editBY,'String',num2str(B(2)))
586 set(handles.editBZ,'String',num2str(B(3)))
587 set(handles.editCX,'String',num2str(C(1)))
588 set(handles.editCY,'String',num2str(C(2)))
589 set(handles.editCZ,'String',num2str(C(3)))
590 set(handles.sliderPX ,'Value',P(1))
591 set(handles.sliderPY ,'Value',P(2))
592 set(handles.sliderPZ ,'Value',P(3))
593 set(handles.sliderQX ,'Value',Q(1))
594 set(handles.sliderQY ,'Value',Q(2))
595 set(handles.sliderQZ ,'Value',Q(3))
596 set(handles.sliderAX ,'Value',A(1))
597 set(handles.sliderAY ,'Value',A(2))
598 set(handles.sliderAZ ,'Value',A(3))
599 set(handles.sliderBX,'Value',B(1))
600 set(handles.sliderBY,'Value',B(2))
601 set(handles.sliderBZ,'Value',B(3))
```

```
set(handles.sliderCX,'Value',C(1))
set(handles.sliderCY,'Value',C(2))
set(handles.sliderCZ,'Value',C(3))
handles.Q=Q; handles.P=P; handles.B=B; handles.A=A; handles.C=C;
PlotBoxAndSegment(handles)
guidata(hObject, handles);


function pushbuttonAB_Callback(hObject, eventdata, handles)
B=handles.A; A=handles.B;
handles.B=B; handles.A=A;
set(handles.editAX ,'String',num2str(A(1)))
set(handles.editAY ,'String',num2str(A(2)))
set(handles.editAZ ,'String',num2str(A(3)))
set(handles.editBX,'String',num2str(B(1)))
set(handles.editBY,'String',num2str(B(2)))
set(handles.editBZ,'String',num2str(B(3)))
set(handles.sliderAX ,'Value',A(1))
set(handles.sliderAY ,'Value',A(2))
set(handles.sliderAZ ,'Value',A(3))
set(handles.sliderBX,'Value',B(1))
set(handles.sliderBY,'Value',B(2))
set(handles.sliderBZ,'Value',B(3))
PlotBoxAndSegment(handles)
guidata(hObject,handles)


function pushbuttonBC_Callback(hObject, eventdata, handles)
C=handles.B; B=handles.C;
handles.C=C; handles.B=B;
set(handles.editCX,'String',num2str(C(1)))
set(handles.editCY,'String',num2str(C(2)))
set(handles.editCZ,'String',num2str(C(3)))
set(handles.editBX,'String',num2str(B(1)))
set(handles.editBY,'String',num2str(B(2)))
set(handles.editBZ,'String',num2str(B(3)))
set(handles.sliderCX,'Value',C(1))
set(handles.sliderCY,'Value',C(2))
set(handles.sliderCZ,'Value',C(3))
set(handles.sliderBX,'Value',B(1))
set(handles.sliderBY,'Value',B(2))
set(handles.sliderBZ,'Value',B(3))
PlotBoxAndSegment(handles)
guidata(hObject,handles)


function pushbuttonCA_Callback(hObject, eventdata, handles)
C=handles.A; A=handles.C;
handles.C=C; handles.A=A;
set(handles.editCX,'String',num2str(C(1)))
set(handles.editCY,'String',num2str(C(2)))
set(handles.editCZ,'String',num2str(C(3)))
set(handles.editAX ,'String',num2str(A(1)))
set(handles.editAY ,'String',num2str(A(2)))
set(handles.editAZ ,'String',num2str(A(3)))
set(handles.sliderCX,'Value',C(1))
set(handles.sliderCY,'Value',C(2))
set(handles.sliderCZ,'Value',C(3))
set(handles.sliderAX ,'Value',A(1))
set(handles.sliderAY ,'Value',A(2))
set(handles.sliderAZ ,'Value',A(3))
PlotBoxAndSegment(handles)
guidata(hObject,handles)
```

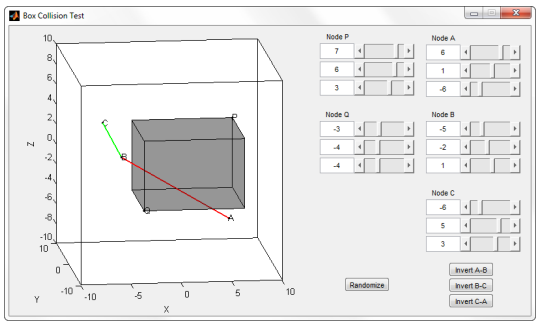
PlotBoxAndSegment.m

```
function []=PlotBoxAndSegment(handles,ColFlag)
```

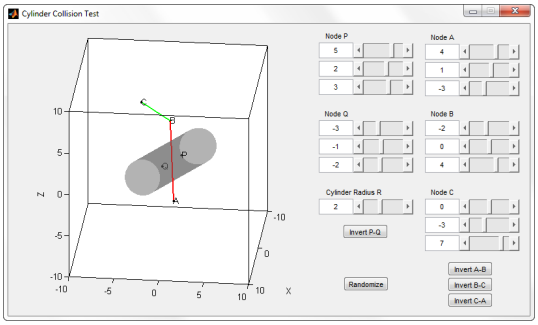
```
2
3  P=handles.P; Q=handles.Q; A=handles.A; B=handles.B; C=handles.C;
4
5  if nargin<2 % If collision flags are not passed, get them
6      ColFlag=rBox(P,Q,[A; B; C],[1 2; 2 3]);
7  end
8
9  axes(handles.axes1), cla, hold on, axis equal, rotate3d on, box on
10 camlight, lighting flat
11
12 % Draw the box using 2 surf calls
13 surf([P(1) Q(1) Q(1) P(1); P(1) Q(1) Q(1) P(1)],...
14      [P(2) P(2) P(2) P(2); Q(2) Q(2) Q(2) Q(2)],...
15      [P(3) P(3) Q(3) Q(3); P(3) P(3) Q(3) Q(3)],0.4*ones(2,4,3))
16 surf([Q(1) P(1) P(1) Q(1); Q(1) P(1) P(1) Q(1)],...
17      [Q(2) Q(2) P(2) P(2); Q(2) Q(2) P(2) P(2)],...
18      [Q(3) Q(3) Q(3) Q(3); P(3) P(3) P(3) P(3)],0.4*ones(2,4,3))
19
20 if ColFlag(1) % If segment 1 collides, draw in red
21     plot3([A(1) B(1)],[A(2) B(2)],[A(3) B(3)],'r','LineWidth',1.5)
22 else % else draw in green
23     plot3([A(1) B(1)],[A(2) B(2)],[A(3) B(3)],'g','LineWidth',1.5)
24 end
25 if ColFlag(2) % If segment 2 collides, draw in red
26     plot3([C(1) B(1)],[C(2) B(2)],[C(3) B(3)],'r','LineWidth',1.5)
27 else % else draw in green
28     plot3([C(1) B(1)],[C(2) B(2)],[C(3) B(3)],'g','LineWidth',1.5)
29 end
30
31 % Plot the control points and label them
32 plot3([P(1) Q(1) A(1) B(1) C(1)],[P(2) Q(2) A(2) B(2) C(2)],[P(3) Q(3) A(3) B(3) C(3)],'k.')
33 text(A(1),A(2),A(3),'A')
34 text(B(1),B(2),B(3),'B')
35 text(C(1),C(2),C(3),'C')
36 text(P(1),P(2),P(3),'P')
37 text(Q(1),Q(2),Q(3),'Q')
38
39 xlabel('X'),ylabel('Y'),zlabel('Z'), axis([-10 10 -10 10 -10 10]), alpha(0.3)
```

rBox.m

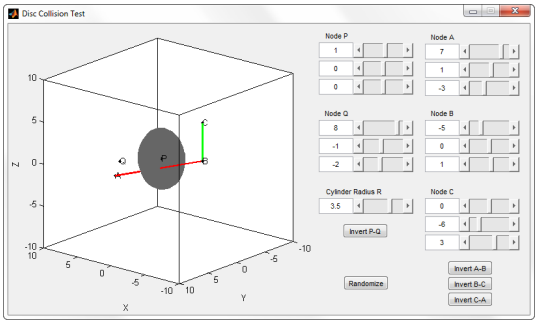
```
1  function flag=rBox(Amin,Amax,NODE,BARS)
2  % Amin and Amax are the box's limit coords: minimum and maximum
3  Nb= size(BARS,1);
4  Tmin = zeros(Nb,1); Tmax = ones(Nb,1);
5  D = NODE(BARS(:,2),:) - NODE(BARS(:,1),:);
6  for i=1:3 % Check all 3 coordinates [X,Y,Z]
7      T1 = ( Amin(i) - NODE(BARS(:,1),i) ) ./ D(:,i);
8      T2 = ( Amax(i) - NODE(BARS(:,1),i) ) ./ D(:,i);
9      ind = find(T1>T2); % We require T1<T2, swap if not
10     [T1(ind),T2(ind)] = deal(T2(ind),T1(ind)); % Swap operation
11     Tmin = max(Tmin,T1); Tmax = min(Tmax,T2);
12 end
13 % No intersection with box if Tmin>Tmax
14 flag = (Tmin<=Tmax);
```



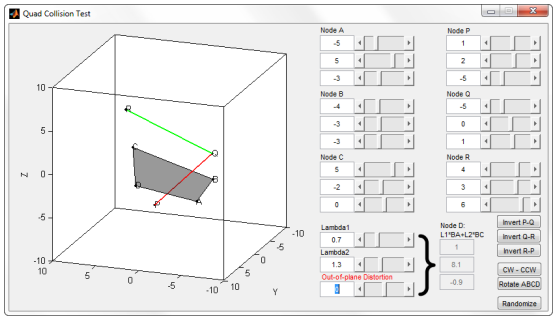
(a)



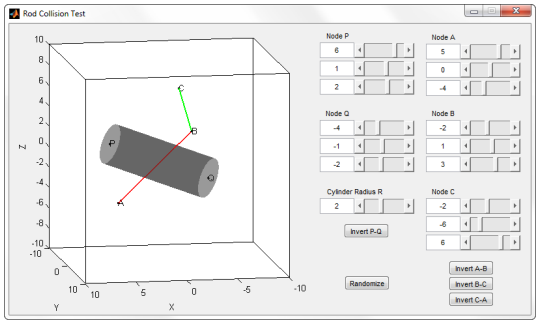
(b)



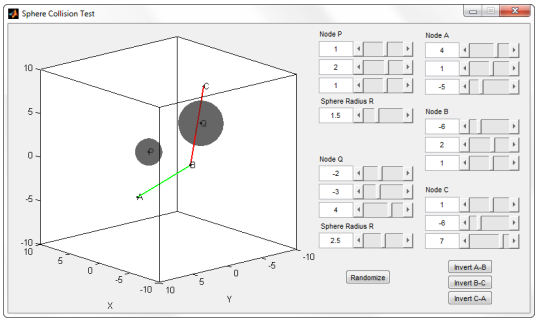
(c)



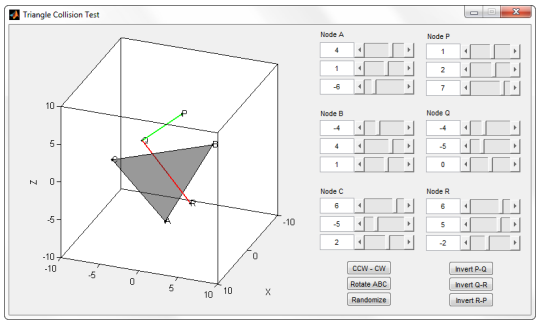
(d)



(e)



(f)



(g)

Figure C.2: Graphical user interfaces to test the collision primitives. (a) Box. (b) Cylinder. (c) Disc. (d) Quadrangle. (e) Rod or finite cylinder. (f) Sphere. (g) Triangle.

References

Aage, N., Andreassen, E., and Lazarov, B. S. (2014). Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, August:1–8. doi: 10.1007/s00158-014-1157-0.

Achtziger, W. (2007). On simultaneous optimization of truss geometry and topology. *Structural and Multidisciplinary Optimization*, 33(4-5):285–304.

ACI Committee (2002). *SP-208: Examples for the Design of Structural Concrete with Strut-and-Tie Models*.

AISC (2011). *Steel construction manual*. Chicago, IL, 14th edition.

Akenine-Möller, T., Haines, E., and Hoffman, N. (2008). *Real-time rendering*. A. K. Peters, Ltd., Natick, MA, 3rd edition.

Allahdadian, S., Boroomand, B., and Barekatein, A. R. (2012). Towards optimal design of bracing system of multi-story structures under harmonic base excitation through a topology optimization scheme. *Finite Elements in Analysis and Design*, 61:60–74.

Allaire, G. and Francfort, G. (1993). A numerical algorithm for topology and shape optimization. In Bendsøe, M. P. and Mota Soares, C. A., editors, *Topology design of structures*, pages 239–248. Springer Netherlands, Sesimbra, Portugal.

Allaire, G. and Kohn, R. (1993). Topology optimization and optimal shape design using homogenization. In Bendsøe, M. P. and Mota Soares, C. A., editors, *Topology design of structures*, pages 207–218. Springer Netherlands, Sesimbra, Portugal.

Almeida, S. R. M., Paulino, G. H., and Silva, E. C. N. (2009). A simple and effective inverse projection scheme for void distribution control in topology optimization. *Structural and Multidisciplinary Optimization*, 39(4):359–371.

Ambrosio, L. and Buttazzo, G. (1993). An optimal design problem with perimeter penalization. *Calculus of Variations and Partial Differential Equations*, 1(1):55–69.

Amir, O. and Sigmund, O. (2013). Reinforcement layout design for concrete structures based on continuum damage and truss topology optimization. *Structural and Multidisciplinary Optimization*, 47(2):157–174.

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., and Sigmund, O. (2011). Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16.

Anstreicher, K. (1999). Linear programming in $\mathcal{O}\left(\frac{n^3}{\ln n}L\right)$ operations. *SIAM Journal on Optimization*, 9(4):803–812.

Baker, W. F. (1992). Energy-based design of lateral systems. *Structural Engineering International*, 2(2):99–102.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483. `http://www.qhull.org/`.

Barzegar, F. and Maddipudi, S. (1994). Generating reinforcement in FE modeling of concrete structures. *Journal of Structural Engineering*, 120(5):1656–1662.

Behr, J., Eschler, P., Jung, Y., and Zöllner, M. (2009). X3DOM: a DOM-based HTML5/X3D integration model. *Proceedings of the 14th International Conference on 3D Web Technology*, pages 127–136. `http://www.x3dom.org/`.

Ben-Tal, A. and Bendsøe, M. P. (1993). A new method for optimal truss topology design. *SIAM Journal on Optimization*, 3(2):322–358.

Bendsøe, M. P. (1989). Optimal shape design as a material distribution problem. *Structural Optimization*, 1(4):193–202.

Bendsøe, M. P. and Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197–224.

Bendsøe, M. P. and Sigmund, O. (1999). Material interpolation schemes in topology optimization. *Archive of Applied Mechanics (Ingenieur Archiv)*, 69(9-10):635–654.

Bendsøe, M. P. and Sigmund, O. (2003). *Topology optimization: theory, methods and applications*. Engineering Online Library. Springer, Berlin, Germany, 2nd edition.

Bourdin, B. (2001). Filters in topology optimization. *International Journal for Numerical Methods in Engineering*, 50(9):2143–2158.

Brackett, D., Ashcroft, I., and Hague, R. (2011). Topology optimization for additive manufacturing. *22nd Annual Solid Freeform Fabrication Symposium*, pages 348–362.

Bruns, T. (2005). A reevaluation of the SIMP method with filtering and an alternative formulation for solid–void topology optimization. *Structural and Multidisciplinary Optimization*, 30(6):428–436.

Bruns, T. E. and Tortorelli, D. A. (2001). Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer Methods in Applied Mechanics and Engineering*, 190(26-27):3443–3459.

Brutzman, D. and Daly, L. (2010). *X3D: extensible 3D graphics for Web authors*. Morgan Kaufmann, San Francisco, CA, USA, 1st edition.

Chakrabarty, J. (2006). *Theory of plasticity*. Butterworth-Heinemann, Oxford, UK, 3rd edition.

Christensen, P. and Klarbring, A. (2009). *An Introduction to Structural Optimization*. Springer, Berlin, Germany, 1st edition.

Crump, S. (1992). Apparatus and method for creating three-dimensional objects. US Patent 5,121,329.

Deaton, J. D. and Grandhi, R. V. (2013). A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization*, 49(1):1–38.

Deckard, C. (1989). Method and apparatus for producing parts by selective sintering. US Patent 4,863,538.

Dewhurst, P. and Srithongchai, S. (2005). An Investigaton of Minimum-Weight Dual-Material Symmetrically Loaded Wheels and Torsion Arms. *Journal of Applied Mechanics*, 72(2):196–202.

Dewhurst, P. and Taggart, D. (2009). Three-dimensional cylindrical truss structures: a case study for topological optimization. In Hernández, S. and Brebbia, C. A., editors, *Computer Aided Optimum Design in Engineering XI*, pages 83–94. WIT Press.

Díaz, A. and Sigmund, O. (1995). Checkerboard patterns in layout optimization. *Structural optimization*, 10(1):40–45.

Dorn, W. S., Gomory, R. E., and Greenberg, H. J. (1964). Automatic design of optimal structures. *Journal de Mecanique*, 3(1):25–52.

Dzierżanowski, G. (2012). On the comparison of material interpolation schemes and optimal composite properties in plane shape optimization. *Structural and Multidisciplinary Optimization*, 46(5):693–710.

Elwi, A. and Hrudey, T. (1989). Finite element model for curved embedded reinforcement. *Journal of engineering mechanics*, 115(4):740–754.

EOS GmbH (accessed June 14, 2014). *Electro Optical Systems: Orthopaedic Technology*. `http://www.eos.info/industries_markets/medical/orthopaedic_technology`.

Ericson, C. (2004). *Real-Time Collision Detection*. Morgan Kaufmann, San Francisco, CA, USA, 1st edition.

Felix, J. and Vanderplaats, G. N. (1987). Configuration optimization of trusses subject to strength, displacement and frequency constraints. *Journal of Mechanical Design*, 109(2):233–241.

France, A. K. (2013). *Make: 3D Printing - The Essential Guide to 3D Printers*. Maker Media, Sebastopol, CA, USA, 1st edition.

Gerdes, D. (1994). *Strukturoptimierung unter Anwendung der Optimalitätskriterien auf diskretisierte Tragwerke bei besonderer Berücksichtigung der Stabilität (in German)*. Phd thesis, Universitłat Essen.

Gilbert, M., Darwich, W., Tyas, A., and Shepherd, P. (2005). Application of large-scale layout optimization techniques in structural engineering practice. *6th World Congress of Structural and Multidisciplinary Optimizartion*, June:1–10.

Gilbert, M. and Tyas, A. (2003). Layout optimization of large-scale pin-jointed frames. *Engineering Computations*, 20(8):1044–1064.

Giles, M. B. and Pierce, N. A. (2000). An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3-4):393–415.

Graczykowski, C. and Lewiński, T. (2005). The lightest plane structures of a bounded stress level transmitting a point load to a circular support. *Control and Cybernetics*, 34(1):227–253.

Graczykowski, C. and Lewiński, T. (2006a). Michell cantilevers constructed within trapezoidal domains—Part I: geometry of Hencky nets. *Structural and Multidisciplinary Optimization*, 32(5):347–368.

Graczykowski, C. and Lewiński, T. (2006b). Michell cantilevers constructed within trapezoidal domains—Part II: virtual displacement fields. *Structural and Multidisciplinary Optimization*, 32(6):463–471.

Graczykowski, C. and Lewiński, T. (2006c). Michell cantilevers constructed within trapezoidal domains—Part III: force fields. *Structural and Multidisciplinary Optimization*, 33(1):1–19.

Graczykowski, C. and Lewiński, T. (2007). Michell cantilevers constructed within trapezoidal domains—Part IV: Complete exact solutions of selected optimal designs and their approximations by trusses of finite number of joints. *Structural and Multidisciplinary Optimization*, 33(2):113–129.

Guest, J. K., Prévost, J. H., and Belytschko, T. (2004). Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *International Journal for Numerical Methods in Engineering*, 61(2):238–254.

Haber, R. B., Jog, C. S., and Bendsøe, M. P. (1996). A new approach to variable-topology shape design using a constraint on perimeter. *Structural Optimization*, 11(1-2):1–12.

Hansen, S. R. and Vanderplaats, G. N. (1990). An approximation method for configuration optimization of trusses. *AAIA Journal*, 28(1):161–168.

Haslinger, J. and Mäkinen, R. (2003). *Introduction to shape optimization: theory, approximation, and computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1st edition.

Heath, M. (1998). *Scientific Computing. An Introductory Survey*. McGraw Hill, New York, NY, USA, 2nd edition.

Hegemier, G. and Prager, W. (1969). On Michell trusses. *International Journal of Mechanical Sciences*, 11(2):209–215.

Hemp, W. S. (1973). *Optimum Structures*. Oxford University Press, Oxford, UK, 1st edition.

Hencky, H. (1923). Über einige statisch bestimmte Fälle des Gleichgewichts in plastischen Körpern. *Z. Angew. Math. Mech*, 747:241–251.

Herceg, M., Kvasnica, M., Jones, C. N., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland. `http://control.ee.ethz.ch/~mpt`.

Hull, C. (1986). Apparatus for production of three-dimensional objects by stereolithography. US Patent 4,575,330.

Imran, I. and Pantazopoulou, S. J. (1996). Experimental study of plain concrete under triaxial stress. *ACI materials Journal*, 93(6):589–601.

Jones, R., Haufe, P., Sells, E., Iravani, P., Olliver, V., Palmer, C., and Bowyer, A. (2011). RepRap — The replicating rapid prototyper. *Robotica*, 29(1):177–191.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

Kato, J. and Ramm, E. (2010). Optimization of fiber geometry for fiber reinforced composites considering damage. *Finite Elements in Analysis and Design*, 46(5):401–415.

Khronos Group (accessed June 16, 2014). *WebGL v1.0 — OpenGL ES 2.0 for the Web*. `http://www.khronos.org/webgl/`.

Kicher, T. P. (1966). Optimum design-minimum weight versus fully stressed. *ASCE Journal of Structural Division*, 92(ST 6):265–279.

Kirsch, U. (1990). On singular topologies in optimum structural design. *Structural Optimization*, 2(3):133–142.

Kirsch, U. (1993). *Structural Optimization: Fundamentals and Applications*. Springer-Verlag, Berlin, Germany, 1st edition.

Lev, O. E. (1981). Topology and optimality of certain trusses. *ASCE Journal of the Structural Division*, 107(ST 2):383–393.

Lewiński, T. (2004). Michell structures formed on surfaces of revolution. *Structural and Multidisciplinary Optimization*, 28(1):20–30.

Lewiński, T. and Rozvany, G. I. N. (2007). Exact analytical solutions for some popular benchmark problems in topology optimization—Part II: three-sided polygonal supports. *Structural and Multidisciplinary Optimization*, 33(4-5):337–349.

Lewiński, T. and Rozvany, G. I. N. (2008a). Analytical benchmarks for topological optimization—Part IV: square-shaped line support. *Structural and Multidisciplinary Optimization*, 36(2):143–158.

Lewiński, T. and Rozvany, G. I. N. (2008b). Exact analytical solutions for some popular benchmark problems in topology optimization—Part III: L-shaped domains. *Structural and Multidisciplinary Optimization*, 35(2):165–174.

Lewiński, T., Rozvany, G. I. N., Sokół, T., and Bołbotowski, K. (2013). Exact analytical solutions for some popular benchmark problems in topology optimization III: L-shaped domains revisited. *Structural and Multidisciplinary Optimization*, 47(6):937–942.

Lewiński, T., Zhou, M., and Rozvany, G. I. N. (1994a). Extended exact least-weight truss layoutsŮ–Part II: unsymmetric cantilevers. *International Journal of Mechanical Sciences*, 36(5):399–419.

Lewiński, T., Zhou, M., and Rozvany, G. I. N. (1994b). Extended exact solutions for least-weight truss layoutsŮ–Part I: cantilever with a horizontal axis of symmetry. *International Journal of Mechanical Sciences*, 36(5):375–398.

Liang, Q. (2007). *Performance-Based Optimization of Structures: Theory and Applications*. Spon Press, New York, NY, USA, 1st edition.

Liang, Q., Xie, Y., and Steven, G. (2000). Optimal topology design of bracing systems for multistory steel frames. *Journal of Structural Engineering*, 127(7):823–829.

Lipson, H. and Kurman, M. (2013). *Fabricated: The new world of 3D printing*. Wiley, Indianapolis, IN, USA, 1st edition.

Lipson, S. L. and Gwin, L. B. (1977). The complex method applied to optimal truss configuration. *Computers & Structures*, 7(3):461–468.

Liu, K. and Tovar, A. (2014). An efficient 3D topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization*, June:1–22. doi: 10.1007/s00158-014-1107-x.

Matsui, K. and Terada, K. (2004). Continuous approximation of material distribution for topology optimization. *International Journal for Numerical Methods in Engineering*, 59(14):1925–1944.

Mazurek, A., Baker, W. F., and Tort, C. (2011). Geometrical aspects of optimum truss like structures. *Structural and Multidisciplinary Optimization*, 43(2):231–242.

Meiners, W., Wissenbach, K., and Gasser, A. (1998). Shaped body especially prototype or replacement part production. DE Patent 19,649,865.

Meisel, N. A., Gaynor, A., Williams, C. B., and Guest, J. K. (2013). Multiple-material topology optimization of compliant mechanisms created via polyjet 3d printing. *24rd Annual International Solid Freeform Fabrication Symposium*, pages 980–997.

Michell, A. G. M. (1904). The limits of economy of material in frame-structures. *Philosophical Magazine Series 6*, 8(47):589–597.

Mijar, A. R., Swan, C. C., Arora, J. S., and Kosaka, I. (1998). Continuum topology optimization for concept design of frame bracing systems. *Journal of Structural Engineering*, 124(5):541–550.

Neves, M. M., Rodrigues, H., and Guedes, J. M. (1995). Generalized topology design of structures with a buckling load criterion. *Structural and Multidisciplinary Optimization*, 10(2):71–78.

Nguyen, T. H., Paulino, G. H., Song, J., and Le, C. H. (2009). A computational paradigm for multiresolution topology optimization (MTOP). *Structural and Multidisciplinary Optimization*, 41(4):525–539.

Nixon, M. and Aguado, A. (2012). *Feature Extraction & Image Processing for Computer Vision*. Elsevier, Oxford, UK, 3rd edition.

Ohsaki, M. (2010). *Optimization of Finite Dimensional Structures*. CRC Press, Boca Raton, FL, USA, 1st edition.

Olson, L. (2013). Personal communication. Department of Computer Science, University of Illinois at Urbana-Champaign.

Petersson, J. (1999). A finite element analysis of optimal variable thickness sheets. *SIAM Journal on Numerical Analysis*, 36(6):1759–1778.

Razani, R. (1965). Behavior of fully stressed design of structures and its relationship to minimum-weight design. *AAIA Journal*, 3(12):2262–2268.

Reinhart, G. and Teufelhart, S. (2011). Load-adapted design of generative manufactured lattice structures. *Physics Procedia*, 12(Part A):385–392.

Rezaie, R., Badrossamay, M., Ghaie, a., and Moosavi, H. (2013). Topology Optimization for Fused Deposition Modeling Process. *Procedia CIRP*, 6:521–526.

Rozvany, G. and Gollub, W. (1990). Michell layouts for various combinations of line supports—I. *International Journal of Mechanical Sciences*, 32(12):1021–1043.

Rozvany, G., Gollub, W., and Zhou, M. (1997). Exact Michell layouts for various combinations of line supports—Part II. *Structural Optimization*, 14(2-3):138–149.

Rozvany, G. I. N. (1996). Some shortcomings in Michell's truss theory. *Structural Optimization*, 12(4):244–250.

Rozvany, G. I. N. (1997a). On the validity of Prager's example of nonunique Michell structures. *Structural optimization*, 13(2-3):191–194.

Rozvany, G. I. N. (1997b). Some shortcomings in Michell's truss theory — Corrigendum. *Structural Optimization*, 13(2-3):203–204.

Rozvany, G. I. N. (1998). Exact analytical solutions for some popular benchmark problems in topology optimization. *Structural optimization*, 15(1):42–48.

Rozvany, G. I. N. (2001). On design-dependent constraints and singular topologies. *Structural and Multidisciplinary Optimization*, 21(2):164–172.

Rozvany, G. I. N. (2009). A critical review of established methods of structural topology optimization. *Structural and Multidisciplinary Optimization*, 37(3):217–237.

Rozvany, G. I. N. and Sokół, T. (2013). *Validation of Numerical Methods by Analytical Benchmarks, and Verification of Exact Solutions by Numerical Methods*. In Topology Optimization in Structural and Continuum Mechanics. Springer, Vienna, Austria.

Rycroft, C. H. (accessed April 21, 2014). Voro++ v0.4.6: a three-dimensional Voronoi cell library in C++. `http://math.lbl.gov/voro++/`.

Salmi, M. (2013). *Medical applications of additive manufacturing in surgery and dental care*. Phd thesis, Aalto University, Helsinki, Finland.

Salmi, M., Tuomi, J., Paloheimo, K.-S., Björkstrand, R., Paloheimo, M., Salo, J., Kontio, R., Mesimäki, K., and Mäkitie, A. A. (2012). Patient-specific reconstruction with 3D modeling and DMLS additive manufacturing. *Rapid Prototyping Journal*, 18(3):209–214.

Schmidt, L. C. (1962). Minimum weight layouts of elastic, statically determinate, triangulated frames under alternative load systems. *Journal of the Mechanics and Physics of Solids*, 10(2):139–149.

Schneider, P. and Eberly, D. (2002). *Geometric tools for computer graphics*. Morgan Kaufmann, San Francisco, CA, USA, 1st edition.

Sigmund, O. (1997). On the Design of Compliant Mechanisms Using Topology Optimization. *Mechanics of Structures and Machines*, 25(4):493–524.

Sigmund, O. (2001). A 99 line topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization*, 21(2):120–127.

Sigmund, O. (2007). Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization*, 33(4-5):401–424.

Sigmund, O. and Maute, K. (2013). Topology optimization approaches. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055.

Sigmund, O. and Petersson, J. (1998). Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 16(1):68–75.

Smith, O. D. S. (1998). Generation of ground structures for 2D and 3D design domains. *Engineering Computations*, 15(4):462–500.

Sokół, T. (2011). A 99 line code for discretized Michell truss optimization written in Mathematica. *Structural and Multidisciplinary Optimization*, 43(2):181–190.

Stolpe, M. and Svanberg, K. (2001). An alternative interpolation scheme for minimum compliance topology optimization. *Structural and Multidisciplinary Optimization*, 22(2):116–124.

Stromberg, L. L., Beghini, A., Baker, W. F., and Paulino, G. H. (2010). Application of layout and topology optimization using pattern gradation for the conceptual design of buildings. *Structural and Multidisciplinary Optimization*, 43(2):165–180.

Stromberg, L. L., Beghini, A., Baker, W. F., and Paulino, G. H. (2012). Topology optimization for braced frames: Combining continuum and beam/column elements. *Engineering Structures*, 37:106–124.

Sundararajan, V. (2011). *Topology optimization for additive manufacturing of customized meso-structures using homogenization and parametric smoothing functions*. Msc thesis, University of Texas at Austin.

Sutradhar, A., Paulino, G. H., Miller, M. J., and Nguyen, T. H. (2010). Topological optimization for designing patient-specific large craniofacial segmental bone replacements. *Proceedings of the National Academy of Sciences of the United States of America*, 107(30):13222–13227.

Svanberg, K. (1987). The method of moving asymptotes - a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373.

Sved, G. (1954). The minimum weight of certain redundant structures. *Australian Journal Of Applied Science*, 5(1):1–9.

Sved, G. and Ginos, Z. (1968). Structural optimization under multiple loading. *International Journal of Mechanical Sciences*, 10(10):803–805.

Talischi, C., Paulino, G. H., Pereira, A., and Menezes, I. F. M. (2012a). PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab. *Structural and Multidisciplinary Optimization*, 45(3):309–328.

Talischi, C., Paulino, G. H., Pereira, A., and Menezes, I. F. M. (2012b). PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. *Structural and Multidisciplinary Optimization*, 45(3):329–357.

Topping, B. H. V. (1983). Shape optimization of skeletal structures: A review. *Journal of Structural Engineering*, 109(8):1933–1951.

Tyas, A., Gilbert, M., and Pritchard, T. (2006). Practical plastic layout optimization of trusses incorporating stability considerations. *Computers & Structures*, 84(3-4):115–126.

Villanueva, C. H. and Maute, K. (2014). Density and level set-XFEM schemes for topology optimization of 3-D structures. *Computational Mechanics*, 54(1):133–150.

Wang, F., Lazarov, B. S., and Sigmund, O. (2011). On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization*, 43(6):767–784.

Wittbrodt, B., a.G. Glover, Laureto, J., Anzalone, G., Oppliger, D., Irwin, J., and Pearce, J. (2013). Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers. *Mechatronics*, 23(6):713–726.

Wright, M. H. (2004). The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56.

Xu, S., Cai, Y., and Cheng, G. (2010). Volume preserving nonlinear density filter based on heaviside functions. *Structural and Multidisciplinary Optimization*, 41(4):495–505.

Zegard, T., Baker, W. F., Mazurek, A., and Paulino, G. H. (2014). Geometrical Aspects of Lateral Bracing Systems: Where Should the Optimal Bracing Point Be? *Journal of Structural Engineering*, 140(9):04014063.

Zegard, T. and Paulino, G. H. (2013a). Toward GPU accelerated topology optimization on unstructured meshes. *Structural and Multidisciplinary Optimization*, 48(3):473–485.

Zegard, T. and Paulino, G. H. (2013b). Truss layout optimization within a continuum. *Structural and Multidisciplinary Optimization*, 48(1):1–16.

Zegard, T. and Paulino, G. H. (2014a). GRAND – Ground structure based topology optimization on arbitrary 2D domains using MATLAB. *Structural and Multidisciplinary Optimization*, 50(5):861–882.

Zegard, T. and Paulino, G. H. (2014b). GRAND3 – Ground structure based topology optimization on arbitrary 3D domains using MATLAB. *Structural and Multidisciplinary Optimization*, Submitted.

Zhang, Y. (1998). Solving large-scale linear programs by interior-point methods under the Matlab Environment. *Optimization Methods and Software*, 10(1):1–31.

Zhou, K. and Li, J. (2005). Forming Michell truss in three-dimensions by Finite Element Method. *Applied Mathematics and Mechanics*, 26(3):381–388.

Zhou, M. and Rozvany, G. I. N. (1991). The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1-3):309–336.