UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Approaches to accommodate remeshing in shape optimization

by

Daniel Nicolas Wilke

A thesis submitted in partial fulfillment
of the requirements for the degree

Philosophiae Doctor (Mechanical Engineering)

in the

Department of Mechanical Engineering
Faculty of Engineering, the Built Environment and Information Technology

University of Pretoria
Pretoria

August 2010

Synopsis

This study proposes novel optimization methodologies for the optimization of problems that reveal non-physical step discontinuities. More specifically, it is proposed to use gradient-only techniques that do not use any zeroth order information at all for step discontinuous problems.

A step discontinuous problem of note is the shape optimization problem in the presence of remeshing strategies, since changes in mesh topologies may — and normally do — introduce non-physical step discontinuities. These discontinuities may in turn manifest themselves as non-physical local minima in which optimization algorithms may become trapped.

Conventional optimization approaches for step discontinuous problems include evolutionary strategies, and design of experiment (DoE) techniques. These conventional approaches typically rely on the exclusive use of zeroth order information to overcome the discontinuities, but are characterized by two important shortcomings: Firstly, the computational demands of zero order methods may be very high, since many function values are in general required. Secondly, the use of zero order information only does not necessarily guarantee that the algorithms will not terminate in highly unfit local minima.

In contrast, the methodologies proposed herein use only first order information, rather than only zeroth order information. The motivation for this approach is that *associated gradient* information in the presence of remeshing remains accurately and uniquely computable, notwithstanding the presence of discontinuities. From a computational effort point of view, a gradient-only approach is of course comparable to conventional gradient-based techniques. In addition, the step discontinuities do not manifest themselves as local minima.

KEYWORDS: shape optimization; gradient-only optimization; unstructured remeshing; truss analogy; analytical sensitivity analysis; consistent tangent; local minima; step discontinuity; partial differential equation; non-constant discretization; error indicator; r-refinement; radial basis function; variable discretization

Sinopsis

Hierdie studie stel 'n nuwe optimerings-metodologie vir die optimering van probleme met nie-fisiese trap diskontinuïteite voor. In besonder word voorgestel om slegs-gradiënt tegnieke, wat glad nie nulde orde inligting benut nie, te gebruik vir trap diskontinue probleme.

'n Trap diskontinue probleem van belang is die vormoptimerings-probleem waar hermaas strategieë gebruik word, omdat veranderinge in maastopologie nie-fisiese trap diskontinuïteite mag veroorsaak. Hierdie diskontinuïteite mag om die beurt weer as nie-fisiese lokale minima te voorskyn kom, waarin optimeringsalgoritmes vasgevang kan raak.

Konvensionele optimeringstegnieke vir trap diskontinue probleme sluit evolutionêre strategieë asook ontwerp van eksperiment (OvE) tegnieke in. Hierdie konvensionele tegnieke maak tipies staat op die uitsluitlike gebruik van nulde orde inligting om diskontinuïteite te oorkom, maar word gekarakteriseer deur twee tekortkominge: Eerstens, die berekeningskoste van nulde orde metodes mag baie hoog wees, omdat baie funksie evaluerings benodig word. Tweedens verseker die gebruik van slegs nulde orde inligting nie dat die algoritmes nie in ongewensde lokale minima termineer nie.

In teenstelling hiermee gebruik die metodologie wat hierin voorgestel word slegs eerste orde inligting, in plaas van nulde orde inligting. Die motivering vir hierdie benadering is dat *geassosieerde gradiënt* inligting in die aanwesigheid van hermasing akkuraat en uniek berekenbaar is, nieteenstaande die teenwoordigheid van diskontinuïteite. Vanuit 'n berekeningsoogpunt is 'n slegs-gradiënt metode natuurlik vergelykbaar met konvensionele gradiënt gebaseerde tegnieke. Boonop manifesteer trap diskonitnuïteite hulself nie as lokale minima nie.

SLEUTELWOORDE: vormoptimering; slegs-gradiënt optimering; ongestruktureerde hermasing; vakwerk-analogie; analitiese sensitiwiteitsanalise; konsekwente gradiënt; lokale minima; stap diskontinuïteit; parsiële differensiaalvergelyking; nie-konstante diskretisering; fout aanwyser; r-verfyning; radiale basis funksie; veranderlike diskretisering

Acknowledgements

I would like to express my sincere appreciation to my supervisors Dr. Schalk Kok and Prof. Albert A. Groendwold for their for guidance, assistance and patience throughout this study. It has been a pleasure and privilege working with you.

To Prof. Jan Snyman I would like to express my sincere gratitude for his guidance, support and motivation of the theoretical section of this study. I will cherish the discussions we have had in your office.

To the support staff of the University of Pretoria, thank you for the professional and friendly support and assistance that you have provided throughout my studies.

This material is based on work supported by the National Research Foundation of South Africa. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Research Foundation.

To the National Research Foundation of South Africa, thank you for make this study financially viable.

Thank you to my beloved father (Piet Wilke) and mother (Brigitte Wilke) for raising me with love and care. Thank you for teaching me the value of education, I have truly been blessed.

To my siblings Johannes Wilke, Dries Wilke and Nina Brazer, thank you for always being there with a supporting shoulder throughout my studies. I look forward to the return the favour at last.

To the teachers whom have taught me throughout my life, from my primary and secondary education at Grey College in Bloemfontein to my tertiary education at the University of the Free State and the University of Pretoria.

A memorable thank you to all my friends from all walks of life, local and abroad, I think of you fondly.

Typeset using LaTeX 2_ε
Compiled under GNU/Linux

Contents

List of Figures

List of Tables

CHAPTER 1

Overview

The following four chapters document the author's contribution as a postgraduate student in the Department of Mechanical and Aeronautical Engineering at the University of Pretoria. Each of the following four chapters is a self-contained advancement towards accommodating remeshing in shape optimization, and are based on published or submitted papers.

In Chapter 2 [66], a novel unstructured remeshing environment for gradient based shape optimisation using triangular finite elements is presented. The remeshing algorithm is based on a truss structure analogy; in solving for the equilibrium position of the truss system, the quadratically convergent Newton's method is used. Analytical sensitivity information of the numerically approximated optimization problem is made available to the shape optimisation algorithm, which results in highly efficient gradient based shape optimisation.

In solving the truss structure analogy in Chapter 2, we compare our quadratically convergent Newton solver with a previously proposed forward Euler solver; this includes notes regarding mesh uniformity, element quality, convergence rates and efficiency. We present three numerical examples; it is then shown that remeshing may introduce discontinuities and local minima. We demonstrate that the effects of these on gradient based algorithms are alleviated to some extent through mesh refinement, and may largely be overcome with a simple multi-start strategy.

In Chapter 3 [65], we study the minimization of objective functions containing non-physical jump discontinuities. These discontinuities arise when (partial) differential equations are discretized using non-constant methods and the resulting numerical solutions are used in computing the objective function, as observed in Chapter 2 using remeshing in shape optimization. Although the functions may become discontinuous and non-differentiable we can compute analytical gradient information of the numerical solution where the function is differentiable, and approximate gradient information where it is discontinuous. At a non-differentiable point a partial derivative of the gradient vector is

constructed by a one-sided directional derivative when the function is respectively non-differentiable or given by the partial derivative itself when the function is differentiable along the partial derivative direction. Such a constructed gradient field follows from the computational scheme since every point has an associated discretization for which sensitivities can be calculated. We refer to this as the *associated gradient* field. Hence, from a computational perspective the *associated gradient* field of these discontinuous functions are everywhere defined albeit approximated at the discontinuities. Rather than the construction of global approximations using only function value information to overcome the discontinuities, as is often done, we propose to use only the *associated gradient* information.

We elaborate on the modifications of classical gradient based optimization algorithms for use in gradient-only approaches, and we then present gradient-only optimization strategies using both BFGS and a new spherical quadratic approximation for sequential approximate optimization (SAO). We also use the BFGS and SAO algorithms to solve three problems of practical interest, both unconstrained and constrained. For the constrained problems we only consider smooth volume constraint functions.

In Chapter 4, we consider some theoretical aspects of gradient-only optimization for the unconstrained optimization of objective functions containing non-physical step or jump discontinuities. The (discontinuous) *associated gradients* are however assumed to be accurate and everywhere uniquely defined. This kind of discontinuity indeed arises when the optimization problem is based on the solution of a system of partial differential equations, when variable discretization techniques are used (remeshing in spatial domains or variable time stepping in temporal domains). These discontinuities, which may cause local minima, are artifacts of the numerical strategies used and should not influence the solution to the optimization problem. We demonstrate that it is indeed possible to ignore these local minima due to discontinuities, if only *associated gradient* information is used. Various gradient-only algorithmic options are discussed. The implications are that variable discretization strategies, so important in the numerical solution of partial differential equations, can be combined with efficient local optimization algorithms.

In Chapter 5, we extend our uniform mesh generator presented in Chapter 2. Herein, we turn our quadratically convergent mesh generator into an adaptive generator, by allowing for a spatially varying ideal element length field, computed using the Zienkiewicz-Zhu error indicator. The remeshing strategy makes (semi) analytical sensitivities available for use in gradient based optimization algorithms. To circumvent difficulties associated with local minima due to remeshing, we again rely on gradient-only optimization algorithms, which do not use zeroth order function information. Numerical results are presented for an orthotropic cantilever beam, an orthotropic Michell-like structure and a spanner design problem.

This study is concluded in Chapter 6, which offers conclusions and recommendations.

CHAPTER 2

Remeshing shape optimization strategy

A novel unstructured remeshing environment for gradient based shape optimization using triangular finite elements is presented. The remeshing algorithm is based on a truss structure analogy; in solving for the equilibrium position of the truss system, the quadratically convergent Newton's method is used. Analytical sensitivity information of the numerically approximated optimization problem is made available to the shape optimization algorithm, which results in highly efficient gradient based shape optimization. In solving the truss structure analogy, we compare our quadratically convergent Newton solver with a previously proposed forward Euler solver; this includes notes regarding mesh uniformity, element quality, convergence rates and efficiency.

We present three numerical examples; it is then shown that remeshing may introduce discontinuities and local minima. We demonstrate that the effects of these on gradient based algorithms are alleviated to some extent through mesh refinement, and may largely be overcome with a simple multi-start strategy.

This chapter is constructed as follows: An outline of shape optimization strategies is given in Section 2.1, followed by the unstructured remeshing strategy is in Section 2.2. In particular, the previously proposed linearly convergent mesh generator based on a truss structure analogy proposed by Persson and Strang is outlined. The mesh generator is then modified to exhibit quadratic convergence in solving for the equilibrium positions of the nodal coordinates of the truss structure. We also present an analytical sensitivity analysis, i.e. the computation of the derivatives of the mesh node positions w.r.t. the design domain control variables. The formulation of the shape design problem is considered in Section 2.3. Finally, numerical results for three example problems are presented in Section 2.4, where after we offer conclusions and recommendations for future work.

2.1 Introduction

Shape optimization involves the constrained minimization of a cost function. The cost function in turn typically involves the solutions of a system of partial differential equations, which depend on parameters that define a geometrical domain [34]. The continuum description of the geometrical domain is normally discretized. This allows efficient solution of the system of partial differential equations, using for example the finite element method (FEM). Normally, the discretized geometric domain is defined by control variables with predefined freedom. The control variables in turn bound the geometrical domain through a predefined relationship, which may be piecewise linear, or based on B-splines, etc.

In shape optimization, different meshing strategies can be used. These include fixed grid strategies [20, 35, 67], design element concepts [28], adaptive mesh strategies [6, 50], and remeshing strategies. The first three methods imply an *a priori* mesh discretization with obvious limitations, for example when dealing with large shape changes in the geometry during optimization. On the other hand, some of the drawbacks of remeshing strategies are the implementation expense, and the possible introduction of local minima, which may cause gradient based optimization methods to become trapped in local minima [1]. However, (unstructured) remeshing strategies allow for generality in structural models and objective functions. Large shape changes can be accommodated using the remeshing strategy with minimal mesh distortion.

In shape optimization, the cost function may be optimized using either a gradient free or gradient based optimization method. While the gradient free methods require only the relationship between the cost function and the discretized geometric domain to be specified, the gradient based optimization methods require additional sensitivity information. The sensitivities needed for the gradient optimization techniques can either be calculated numerically, semi-analytically or analytically. All these methods have merits and drawbacks. Numerical gradients using finite difference methods are computationally expensive, but are easily implementable. The semi-analytical and analytical methods are more complex to implement, but are computationally cheaper.

An advantage of gradient free evolutionary strategies is their global optimization capability. They have been used with success by Xie and Steven [35, 67] in a fixed grid strategy. Related works that reflect evolutionary strategies in shape optimization, are the biological growth method of Mattheck and Burkhardt [38], and the genetic algorithm used by Garcia and Gonzalez [20]. They are in general however, still very expensive, and the solutions are normally inferior to those obtained with gradient based methods. Most certainly so for problems with many design variables. Hence we restrict ourselves to gradient based methods in this study.

Mesh generation plays an important role in shape optimization and in general con-

tributes largely to the computational expense per iteration when unstructured remeshing strategies are used. This cost may however be offset many times over when exact analytical gradients can be made available to the algorithm used in solving the shape optimization problem. Remeshing strategies in shape optimization accentuate the importance of robustness, computational speed, flexibility and accuracy of the mesh generator in discretizing the geometrical domain.

In this study a novel remeshing shape optimization environment is presented. The environment is based on an elegant truss structure analogy proposed by Persson and Strang [42]. It is however developed such that the analytical sensitivities are available, cost effectively. Two gradient based optimization algorithms are implemented, namely the Dynamic-Q algorithm [56], and sequential quadratic programming (SQP) [4]. These algorithms are then used to solve example problems in shape optimization. In turn, this demonstrates that remeshing may introduce discontinuities, which may cause the gradient based algorithms to become trapped in local minima. We then investigate the ability of h-refinement to escape from these local minima.

2.2 Mesh generation

Computational meshes are used extensively in engineering and physics to discretize a continuous geometrical domain Ω with boundary $\partial\Omega$. The computational mesh

$$\Lambda \in \{\boldsymbol{\mathcal{X}} = (\boldsymbol{X}_i)_{i=1,\ldots,nn}; T = (T_j^k)_{j=1,\ldots,ne;k=1,\ldots,nv}\}, \tag{2.1}$$

defined on the domain Ω describes the position $\boldsymbol{\mathcal{X}} \in \mathbb{R}^3$ of the nn nodes, and gives for each of the $j = 1, \ldots, ne$ computational elements the set $T_j^{k=1,\ldots,nv}$ of its nv vertices [34]. In addition the set of nodes $\boldsymbol{\mathcal{X}}$ are the union of the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ and the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$.

In this study we limit the nodal positions to two dimensions $\boldsymbol{\mathcal{X}} \in \mathbb{R}^2$. In this section, we present triangulation based on the truss structure analogy proposed by Persson and Strang [42]. This incorporates a Delaunay strategy [18] to ensure good mesh quality, albeit at the cost of potentially introducing discontinuities between consecutive meshes due to the addition or removal of nodes.

2.2.1 Mesh generator based on a truss structure analogy

The mesh generator proposed by Persson and Strang is based on a truss structure analogy that solves for the equilibrium position of a truss structure. The geometrical domain is defined by a signed distance function that signs the nodes outside the domain as positive, inside as negative and zero on the boundary. The distance function is a function of the

control variables through the interpolation of the domain. The initial mesh is generated using the simple algorithm of Persson and Strang, which mostly creates equilateral triangles in the domain.

The truss force function F is defined with a force discontinuity, as no tensile forces are permitted in the truss elements. This allows the propagation of the nodes $\boldsymbol{\mathcal{X}}$ to the boundary $\partial\Omega$. The nodes are kept inside the geometrical domain by external forces acting on the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$. The forces act perpendicularly to the boundary, keeping the nodes from moving outside the boundary while allowing movement along the boundary.

The truss force function F is defined as

$$F(l, h_0) = \begin{cases} k(h_0 - l) & \text{if } l < h_0 \\ 0 & \text{if } l \geq h_0 \end{cases} \tag{2.2}$$

with k the spring (truss) stiffness, l the current spring length and h_0 the undeformed spring length (also referred to as the ideal element length). The undeformed spring length h_0 is a user specified parameter whereas the current spring length $l(\boldsymbol{\mathcal{X}})$ is a function of the nodal positions $\boldsymbol{\mathcal{X}}$. The nodal positions $\boldsymbol{\mathcal{X}}(\boldsymbol{x})$ in turn depends on the control variables $\boldsymbol{x}$. There is also a dependency of l on the mesh topology T, which we omit since the mesh topology T converges to a constant topology as the equilibrium of the truss structure converges. The implication is the introduction of discontinuities in the residual of the equilibrium of the truss structure whenever T or $\boldsymbol{\mathcal{X}}$ changes, due to Delaunay triangulation. In the implementation of Persson and Strang [42], all springs are precompressed by 20%, which provides the driving force necessary to propagate nodes to the boundary.

The truss system $\boldsymbol{F}(\boldsymbol{\mathcal{X}}) = \boldsymbol{0}$ is transformed to a system of ordinary differential equations through the introduction of artificial time-dependence in the equations. The system is then solved by a forward Euler method

$$\boldsymbol{\mathcal{X}}_{n+1} = \boldsymbol{\mathcal{X}}_n + \Delta t \boldsymbol{F}(\boldsymbol{\mathcal{X}}_n). \tag{2.3}$$

The forward Euler method is essentially a matrix free method ideally suited to create meshes with a very large number of elements. This method exhibits linear convergence rates.

However, in general, the structural meshes (number of elements) in shape optimization tend to vary from small to moderate for practical optimization problems, since optimization is *per se* computationally expensive. Emphasis is placed on mesh quality and accurate representation of the geometrical domain. It may therefore be beneficial from a computational cost perspective to replace the forward Euler method with a quadratically convergent scheme. We will do so in the next subsection.

2.2.2 Quadratic convergent Newton solver for the mesh generator

The truss system equilibrium equations $\boldsymbol{F}(\boldsymbol{\mathcal{X}}) = \boldsymbol{0}$ are partitioned along the internal nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ and boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ i.e.

$$\boldsymbol{F}(\boldsymbol{\mathcal{X}}) = \left\{ \begin{array}{c} \boldsymbol{F}_{\Omega}(\boldsymbol{\mathcal{X}}) \\ \boldsymbol{F}_{\partial\Omega}(\boldsymbol{\mathcal{X}}) \end{array} \right\} = \left\{ \begin{array}{c} \boldsymbol{F}_{\Omega}(\boldsymbol{\mathcal{X}}^{\Omega}, \boldsymbol{\mathcal{X}}^{\partial\Omega}) \\ \boldsymbol{F}_{\partial\Omega}(\boldsymbol{\mathcal{X}}^{\Omega}, \boldsymbol{\mathcal{X}}^{\partial\Omega}) \end{array} \right\} = \left\{ \begin{array}{c} \boldsymbol{0} \\ \boldsymbol{0} \end{array} \right\}. \tag{2.4}$$

For the sake of simplicity, the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ are seeded along the geometrical boundary $\partial\Omega$ and are chosen to remain fixed during the shape mesh generation process. Nodes are explicitly placed on the control variable locations to ensure accurate representation of the defined geometrical domain Ω. Since the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ are fixed, the system of unknowns reduces to $\boldsymbol{\mathcal{X}}^{\Omega}$. Hence, we rewrite $\boldsymbol{F}_{\Omega}(\boldsymbol{\mathcal{X}}^{\Omega}, \boldsymbol{\mathcal{X}}^{\partial\Omega})$ as $\boldsymbol{F}_{\Omega}(\boldsymbol{\mathcal{X}}^{\Omega})$. Also, the reactions at the boundary nodes are not of immediate interest, so we only solve for

$$\boldsymbol{F}_{\Omega}(\boldsymbol{\mathcal{X}}^{\Omega}) = \boldsymbol{0}. \tag{2.5}$$

The reduced truss system in Eq. (2.5) is solved directly via the quadratically convergent Newton's method, i.e. we solve for $\Delta\boldsymbol{\mathcal{X}}^{\Omega}$ from

$$\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} \Delta\boldsymbol{\mathcal{X}}^{\Omega} = -\boldsymbol{F}_{\Omega} \tag{2.6}$$

to update the nodal coordinates

$$\boldsymbol{\mathcal{X}}^{\Omega}_{n+1} = \boldsymbol{\mathcal{X}}^{\Omega}_{n} + \Delta\boldsymbol{\mathcal{X}}^{\Omega}. \tag{2.7}$$

The consistent tangent $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}}$ is computed analytically for every iteration, since the number of elements (and hence the number of unknowns) and element connectivity may change between consecutive iterations, due to Delaunay triangulation. Although unusual, the possible change in the number of system unknowns requires no special treatment.

Since the force function F proposed by Persson and Strang in Eq. (2.2) is discontinuous, (which is undesirable in gradient based implementations), it is now changed to allow for both tensile and compressive forces in the truss elements in finding the equilibrium position, i.e.

$$F\left(l(\boldsymbol{\mathcal{X}}(\boldsymbol{x})), h_0\right) = k(h_0 - l(\boldsymbol{\mathcal{X}}(\boldsymbol{x}))) \qquad \text{for all } l. \tag{2.8}$$

Furthermore, we do not require any precompression in the springs.

Table 2.1: Cartesian coordinates for the piece-wise linear boundary description of the circular part of the quarter circular disc.

x	0	2.9264	5.7403	8.3336	10.6066	12.472	13.8582	14.7118	15
y	15	14.7118	13.8582	12.472	10.6066	8.3336	5.7403	2.9264	0

2.2.3 Evaluation of the mesh generators

We now compare our novel quadratically convergent Newton solver to the forward Euler solver. The boundary nodes are treated as discussed in Section 2.2.2 for both the Newton and forward Euler implementations.

A comparative study is done for the mesh generation of a quarter circular disk with a radius of 15 units. The x and y coordinates for the piece-wise linear boundary representation of the circular part of the quarter circular disk is given in Table 2.1. The comparison focuses on the computational expense and the convergence rate of both solvers. (In cases where the sensitivities are not needed, i.e. for gradient free optimization methods, implementations of Quasi-Newton or Modified Newton methods can be used to obtain a computational advantage. Additionally, the residual convergence tolerance may also be relaxed.)

A disadvantage of the Newton solver is that matrix methods require extensive memory resources when the mesh size is increased, when compared to the matrix-free forward Euler solver. However, we utilise sparse matrix manipulation techniques whenever possible.

For both the forward Euler and Newton methods, we express the stopping condition in terms of the maximum nodal displacement, i.e. we stop when $|\boldsymbol{\mathcal{X}}_{n+1} - \boldsymbol{\mathcal{X}}_n|_\infty < \epsilon$, with $\epsilon > 0$, small and prescribed. This stopping criterion was also used by Persson and Strang [42]. The study is conducted using a 3GHz Pentium IV machine with 512 MB RAM running under the Linux operating system.

Figure 2.1 depicts the computational effort comparison for the Newton and forward Euler solvers for different mesh sizes. For a stopping tolerance of $\epsilon = 0.04h_0$, the computational expense of the forward Euler method is comparable to Newton's method, where we use a stopping tolerance of $\epsilon = 10^{-8}h_0$. However, decreasing the stopping tolerance for the forward Euler solver by a factor 10 increases the computational effort on average by a factor of 16.

Figure 2.2 depicts the force residual versus the number of iterations for both solvers using an ideal element length of $h_0 = 0.375$. After the mesh stabilises the convergence rate is quadratic for the Newton solver. (We define mesh stability to imply that no elements are added or removed from the mesh from one iteration to the next as a result of Delaunay

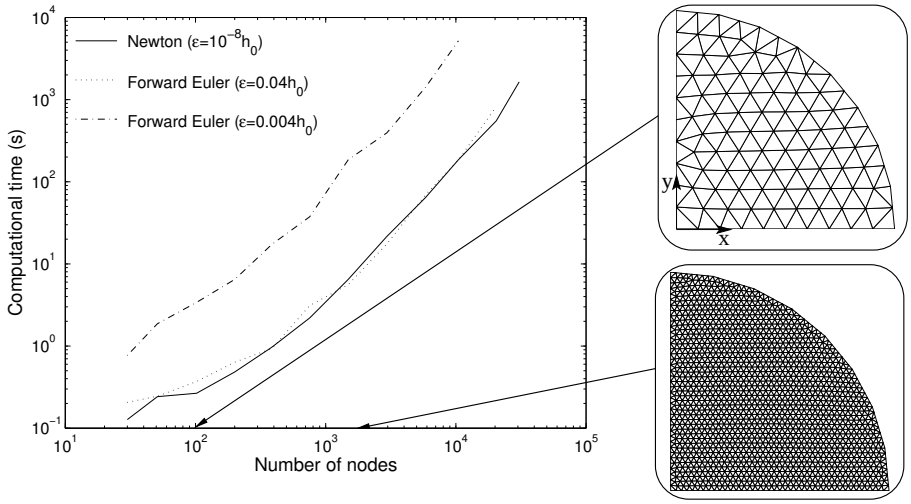


Figure 2.1: Computational effort required to solve the truss equilibrium with the Newton and the forward Euler methods.

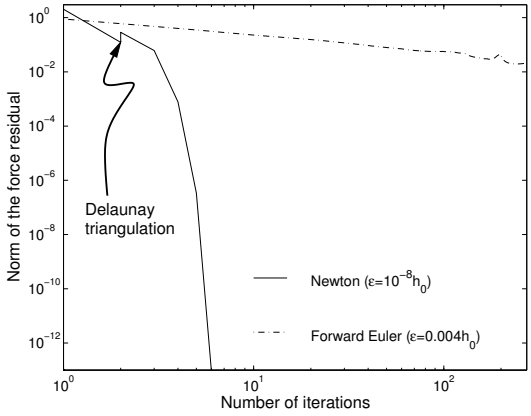
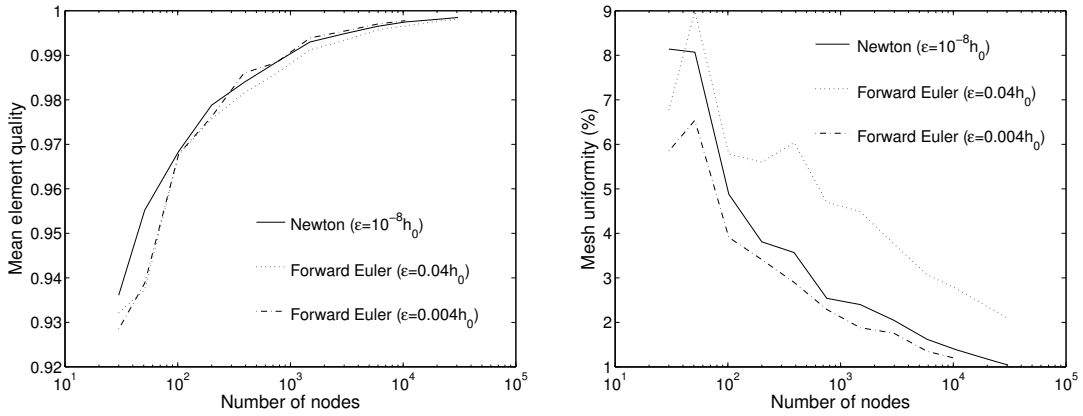


Figure 2.2: Force residual comparison for an ideal element length of $h_0 = 0.375$ with 1492 nodes.

triangulation.) In general, this requires between 1-2 iterations. The discontinuity in the force residual visible after the first iteration in Figure 2.2 is an example of such an event. (Even including these events, Newton's method requires only 6 iterations on average for all meshes we constructed.) Also illustrated is the linear convergence rate of the forward Euler solver. For this solver the average number of iterations increases from 10 to 155 as the tolerance is decreased from $0.04h_0$ to $0.004h_0$. For our implementation however, the average computational cost per iteration of the forward Euler solver is some 30% less than the average computational cost for the Newton solver.

The forward Euler implementation is different from our implementation since it does not allow tensile forces in the trusses and a precompression is imposed. It is therefore necessary to verify that the changes we made are not detrimental to the element quality and mesh uniformity reported in [42].

Element quality is defined as twice the ratio of the radius of the largest inscribed



(a) (b)

Figure 2.3: Comparison between the different solvers in terms of a) mean element quality and b) mesh uniformity for the quarter circular disc.

Table 2.2: Minimum element qualities for the forward Euler and Newton solvers.

h_0	forward Euler ($\epsilon = 0.04h_0$)	forward Euler ($\epsilon = 0.004h_0$)	Newton ($\epsilon = 10^{-8}h_0$)
3	0.7942	0.8197	0.7435
1.5	0.7323	0.7384	0.6915
0.75	0.6312	0.7644	0.6665
0.375	0.5845	0.7191	0.6571
0.1875	0.6245	0.7625	0.6538

circle over the radius of the smallest circumscribed circle. Hence the element quality of an equilateral triangle for example is 1.00, while the element quality of a 30-60-90 angle triangle is only 0.68.

Mesh uniformity is defined as the standard deviation of the ratio of the circumradii of all the triangles in the mesh to the ideal element length h_0. The mesh uniformity is normalized by the mean ratio, and then expressed as a percentage. Hence a mesh uniformity of 0% is the ideal.

We compare the mean element quality and mesh uniformity of the two solvers in Figures 2.3(a) and 2.3(b) respectively. For this comparison we again use the quarter circular disc. In essence, the mean element quality and the mesh uniformity are similar. In Table 2.2, the lowest element quality in the meshes are compared. Again, the solvers perform comparable.

2.3 Problem formulation

In general the shape optimization problem is given in the following abstract form [34].

Problem 2.3.1. *Find the optimal shape* Ω^* *such that*

$$\mathcal{F}_\Omega^* = \mathcal{F}_\Omega\big(\Omega^*, S_\Omega^*\big) = \min_\Omega\{\mathcal{F}_\Omega\big(\Omega, S_\Omega\big) : G\big(\Omega\big) \leq \epsilon_0\}, \tag{2.9}$$

with S_Ω *the state solution of the partial differential equations* $p_\Omega(\Omega, S_\Omega) = 0$*, which characterizes the system [34].*

Here, Ω denotes the unknown shape, which is a subset in $\mathbb{R}^3$. Ω is usually defined by a finite set of control variables $\boldsymbol{x} \in \mathbb{R}^n$ and $G(\Omega) \leq \epsilon_0$ the set of nonlinear design constraints, with ϵ_0 given.

Problem 2.3.1 is ill-posed but can be reduced to an approximate problem by discretizing the domain Ω with a computational grid Λ (see Eq. (2.1)). The computational grid Λ can then be used to approximate the state equation $p_\Lambda(\Lambda, S_\Lambda) = 0$ and to define an approximate solution S_Λ to the state equation. The set of nonlinear design constraints $G\big(\Omega\big) \leq \epsilon_0$ can then also be expressed as m functions of the computational grid Λ as follows

$$g_i\big(\Lambda(\boldsymbol{x}), \boldsymbol{x}\big) \leq 0, \quad i = 1, \ldots, m. \tag{2.10}$$

Consequently the shape optimization problem reduces to the following approximate problem.

Problem 2.3.2. *Find the minimum* $\mathcal{F}^*$ *such that*

$$\mathcal{F}^* = \mathcal{F}\big(\Lambda^*(\boldsymbol{x}^*), S_\Lambda^*(\boldsymbol{x}^*)\big) = \min_{\boldsymbol{x}\in\mathbb{R}^n}\{\mathcal{F}\big(\Lambda(\boldsymbol{x}), S_\Lambda(\boldsymbol{x})\big) : \boldsymbol{g}\big(\Lambda(\boldsymbol{x}), \boldsymbol{x}\big) \leq \boldsymbol{0}\}, \tag{2.11}$$

with S_Λ *the approximate state solution of the approximate partial differential equations* $p_\Lambda(\Lambda, S_\Lambda) = 0$*, which characterizes the system.*

For the sake of brevity, the objective function and the constraints will respectively be denoted by $\mathcal{F}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$; this notation will however imply dependency on $\Lambda(\boldsymbol{x})$. The objective and constraint functions can be selected in many ways. In structural shape optimization the objective function is usually chosen as the weight or volume of the structure, subject to displacement and stress constraints [16].

In our case, the objective function $\mathcal{F}(\boldsymbol{x}) = \mathcal{F}\big(\boldsymbol{u}(\Lambda(\boldsymbol{x}))\big)$ is an explicit function of the nodal displacements $\boldsymbol{u}$, which are obtained by solving the approximate finite element equilibrium equations for linear elasticity, formulated as

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{2.12}$$

where $\boldsymbol{K}$ represents the assembled structural stiffness matrix and $\boldsymbol{f}$ the consistent structural loads. Following the usual approach, the system in Eq. (2.12) is partitioned along

the unknown displacements ($\boldsymbol{u}_\mathrm{f}$) and the prescribed displacement ($\boldsymbol{u}_\mathrm{p}$), i.e.

$$\boldsymbol{K}\boldsymbol{u} = \begin{bmatrix} \boldsymbol{K}_\mathrm{ff} & \boldsymbol{K}_\mathrm{fp} \\ \boldsymbol{K}_\mathrm{pf} & \boldsymbol{K}_\mathrm{pp} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_\mathrm{f} \\ \boldsymbol{u}_\mathrm{p} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_\mathrm{f} \\ \boldsymbol{f}_\mathrm{p} \end{Bmatrix}, \tag{2.13}$$

where $\boldsymbol{f}_\mathrm{f}$ represents the prescribed forces and $\boldsymbol{f}_\mathrm{p}$ the reactions at the nodes with prescribed displacements. The unknown displacements ($\boldsymbol{u}_\mathrm{f}$) are obtained from

$$\boldsymbol{K}_\mathrm{ff}\boldsymbol{u}_\mathrm{f} = \boldsymbol{f}_\mathrm{f} - \boldsymbol{K}_\mathrm{fp}\boldsymbol{u}_\mathrm{p}. \tag{2.14}$$

We choose to represent the geometrical domain boundary $\partial\Omega$ by a simple piecewise linear interpolation between the control variables. However, Bezier curves or B-splines, etc. may of course also be used. We subject the structure to a volume constraint $g(\boldsymbol{x})$ which reduces to a linear function of the control variables $\boldsymbol{x}$.

2.3.1 Analytical sensitivities

Recall that the cost function $\mathcal{F}(\boldsymbol{x})$ is an explicit function of the nodal displacements $\boldsymbol{u}$. Using gradient based optimization algorithms, we therefore require the sensitivity of the structural response $\boldsymbol{u}$ w.r.t. the design variables (control variables) $\boldsymbol{x}$. In general, the stiffness partition matrices $\boldsymbol{K}_\mathrm{ff}$ and $\boldsymbol{K}_\mathrm{fp}$, the nodal displacement vector $\boldsymbol{u}_\mathrm{f}$ and the load vector $\boldsymbol{f}_\mathrm{f}$ in Eq. (2.14) depend on the design variables $\boldsymbol{x}$, i.e. $\boldsymbol{K}_\mathrm{ff}(\boldsymbol{x})\boldsymbol{u}_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{f}_\mathrm{f}(\boldsymbol{x}) - \boldsymbol{K}_\mathrm{fp}(\boldsymbol{x})\boldsymbol{u}_\mathrm{p}(\boldsymbol{x})$.

The analytical gradient $\frac{d\boldsymbol{u}_\mathrm{f}}{d\boldsymbol{x}}$ is obtained by differentiating Eq. (2.14) w.r.t. the control variables $\boldsymbol{x}$, i.e.

$$\boldsymbol{K}_\mathrm{ff}\frac{d\boldsymbol{u}_\mathrm{f}}{d\boldsymbol{x}} = \frac{d\boldsymbol{f}_\mathrm{f}}{d\boldsymbol{x}} - \frac{d\boldsymbol{K}_\mathrm{fp}}{d\boldsymbol{x}}\boldsymbol{u}_\mathrm{p} - \boldsymbol{K}_\mathrm{fp}\frac{d\boldsymbol{u}_\mathrm{p}}{d\boldsymbol{x}} - \frac{d\boldsymbol{K}_\mathrm{ff}}{d\boldsymbol{x}}\boldsymbol{u}_\mathrm{f}. \tag{2.15}$$

In this study the load vector $\boldsymbol{f}_\mathrm{f}$ is assumed to be independent of the control variables $\boldsymbol{x}$, hence $\frac{d\boldsymbol{f}_\mathrm{f}}{d\boldsymbol{x}} = \boldsymbol{0}$. For Dirichlet boundary conditions, $\boldsymbol{u}_\mathrm{p} = \boldsymbol{0}$, and Eq. (2.15) reduces to

$$\boldsymbol{K}_\mathrm{ff}\frac{d\boldsymbol{u}_\mathrm{f}}{d\boldsymbol{x}} = -\frac{d\boldsymbol{K}_\mathrm{ff}}{d\boldsymbol{x}}\boldsymbol{u}_\mathrm{f}. \tag{2.16}$$

Eq. (2.16) is solved to obtain $\frac{d\boldsymbol{u}_\mathrm{f}}{d\boldsymbol{x}}$, using the factored stiffness matrix $\boldsymbol{K}_\mathrm{ff}$, available from the primary analysis when solving Eq. (2.14). The unknown $\frac{d\boldsymbol{K}_\mathrm{ff}}{d\boldsymbol{x}}$ is computed from

$$\frac{d\boldsymbol{K}_\mathrm{ff}}{d\boldsymbol{x}} = \frac{d\boldsymbol{K}_\mathrm{ff}}{d\boldsymbol{\mathcal{X}}}\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}, \tag{2.17}$$

where $\frac{d\boldsymbol{K}_\mathrm{ff}}{d\boldsymbol{\mathcal{X}}}$ is obtained by differentiating the stiffness matrix analytically with respect to the nodal coordinates $\boldsymbol{\mathcal{X}}$. This is done on the element level and then assembled into the global system. For simplicity's sake we choose to use the constant strain triangle (CST)

element. The element stiffness matrix $\boldsymbol{K}^e$ of the CST element is given by

$$\boldsymbol{K}^e = tA\boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}, \tag{2.18}$$

where t and A denote the element thickness and element area. Hence $\frac{d\boldsymbol{K}^e}{d\boldsymbol{\mathcal{X}}}$ is given by

$$\frac{d\boldsymbol{K}^e}{d\boldsymbol{\mathcal{X}}} = t\left(\frac{dA}{d\boldsymbol{\mathcal{X}}}\boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B} + A\frac{d\boldsymbol{B}^T}{d\boldsymbol{\mathcal{X}}}\boldsymbol{D}\boldsymbol{B} + A\boldsymbol{B}^T\boldsymbol{D}\frac{d\boldsymbol{B}}{d\boldsymbol{\mathcal{X}}}\right). \tag{2.19}$$

The area of the CST element is given by

$$A = 0.5|\det\boldsymbol{J}| = 0.5|x_{13}y_{23} - x_{23}y_{13}|. \tag{2.20}$$

Here, $\boldsymbol{J}$ represents the Jacobian matrix and $x_{ij} = x_i - x_j$, etc. Subscripts $i, j = 1, 2, 3$ denote the element node numbers. The CST element strain-displacement matrix is

$$\boldsymbol{B} = \frac{1}{\det\boldsymbol{J}}\begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}. \tag{2.21}$$

It follows from Eqs. (2.18), (2.20) and (2.21) that $\boldsymbol{K}^e$ is a nonlinear function of the nodal coordinates $\boldsymbol{\mathcal{X}}$. $\boldsymbol{B}$ and A are differentiated directly to obtain $\frac{d\boldsymbol{K}^e}{d\boldsymbol{\mathcal{X}}}$; assembly yields $\frac{d\boldsymbol{K}_{\text{ff}}}{d\boldsymbol{\mathcal{X}}}$.

To complete the sensitivity analysis, we still need to evaluate $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ present in Eq. (2.17). To emphasize the dependency on the control variables $\boldsymbol{x}$, the reduced truss system $\boldsymbol{F}_\Omega$ is now expressed as a function of both the interior nodes $\boldsymbol{\mathcal{X}}^\Omega(\boldsymbol{x})$ and the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}(\boldsymbol{x})$, i.e.

$$\boldsymbol{F}_\Omega(\boldsymbol{\mathcal{X}}^\Omega(\boldsymbol{x}), \boldsymbol{\mathcal{X}}^{\partial\Omega}(\boldsymbol{x})) = \boldsymbol{0}. \tag{2.22}$$

To determine the relationship between the nodal coordinates $\boldsymbol{\mathcal{X}}$ and the control variables $\boldsymbol{x}$, we take the derivative of Eq. (2.22) w.r.t. $\boldsymbol{x}$, i.e.

$$\frac{d\boldsymbol{F}_\Omega}{d\boldsymbol{x}} = \frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^\Omega}\frac{d\boldsymbol{\mathcal{X}}^\Omega}{d\boldsymbol{x}} + \frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\boldsymbol{x}} = \boldsymbol{0} \tag{2.23}$$

hence

$$\frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^\Omega}\frac{d\boldsymbol{\mathcal{X}}^\Omega}{d\boldsymbol{x}} = -\frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\boldsymbol{x}}. \tag{2.24}$$

$\frac{d\boldsymbol{\mathcal{X}}^\Omega}{d\boldsymbol{x}}$ can be obtained if $\frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\boldsymbol{x}}$ and $\frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^\Omega}$ are known, either analytically or numerically. Although the relationship between the control variables $\boldsymbol{x}$ and the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ is known explicitly due to the linear relationship of our piece-wise linear boundary, we compute $\frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\boldsymbol{x}}$ using a semi-analytical sensitivity analysis [40]. $\frac{\partial\boldsymbol{F}_\Omega}{\partial\boldsymbol{\mathcal{X}}^\Omega}$ is available from

Newton's method (cf. Eq. (2.6)) implemented in the mesh generation step. Therefore we obtain $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ as the union of $\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\boldsymbol{x}}$ and $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\boldsymbol{x}}$. The semi-analytical sensitivity calculation requires finite difference perturbations from an unperturbed geometry. For the unperturbed and perturbed geometries we ensure the mesh topology T remains the same by deactivating the Delaunay triangulation for the duration of the sensitivity calculation. This avoids remeshing between the unperturbed and perturbed geometries and any inconsistencies in mesh topology that may occur.

In summary, the primary and sensitivity analyses proceed as follows:

1. Solve for the nodal positions $\boldsymbol{\mathcal{X}}^{\Omega}$ by solving Eq. (2.6) repeatedly. $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}}$ and $\boldsymbol{F}_{\Omega}$ are recomputed at each iteration.

2. Calculate $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega}} \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\boldsymbol{x}}$ semi-analytically and then solve for $\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\boldsymbol{x}}$ from Eq. (2.24), using $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}}$ from step 1 above.

3. Assemble $\boldsymbol{K}_{\mathrm{ff}}$ and $\boldsymbol{f}_{\mathrm{f}}$, then solve for $\boldsymbol{u}_{\mathrm{f}}$ from Eq. (2.14).

4. Compute $\frac{d\boldsymbol{K}^{e}}{d\boldsymbol{\mathcal{X}}}$ using Eq. (2.19) and assemble over all elements to obtain $\frac{d\boldsymbol{K}_{\mathrm{ff}}}{d\boldsymbol{\mathcal{X}}}$.

5. Compute $\frac{d\boldsymbol{K}_{\mathrm{ff}}}{d\boldsymbol{x}}$ using Eq. (2.17), and using $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ from step 2 above.

6. Solve for $\frac{d\boldsymbol{u}_{\mathrm{f}}}{d\boldsymbol{x}}$ using Eq. (2.16).

2.3.2 Gradient sensitivity comparison

To verify that no errors were made during the analytical gradient derivations, we compare our analytical sensitivities to numerical sensitivities obtained with the forward finite difference method. We use the bow-tie structure and mesh depicted in Figure 2.4 to compute the sensitivity of the displacement at the point of load application (u_{F}) w.r.t. the indicated control variables. The x and y coordinates for control variables and applied load F of the the bow-tie structure of the unperturbed structure are given in Table 2.3.

Calculation of the numerical sensitivities is conducted without Delaunay triangulation steps, to avoid the introduction of any discontinuity (due to the addition or removal of

Table 2.3: Cartesian coordinates for control variables and applied load F of the unperturbed bow-tie structure.

Control variable	1	2	3	4	5	6	7	8	F
x	5	10	15	20	5	10	15	20	20
y	15	9	15	15	0	6	0	0	7.5

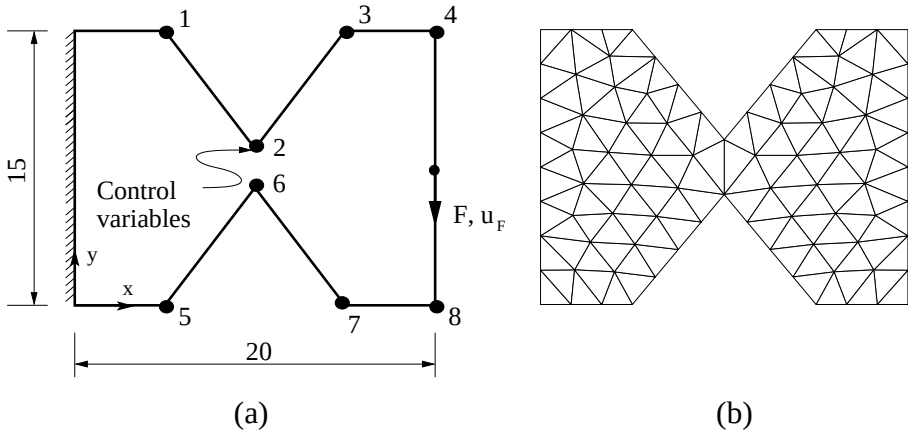


Figure 2.4: a) Bow-tie structure defined by the indicated control variables and b) the associated mesh for an ideal element length of $h_0 = 2.0$.

nodes) in the numerical sensitivity analysis. In Table 2.4 the numerical gradients for a perturbation of 10^{-6} are compared to the analytical gradients. It follows from Table 2.4 that our computations are correct and accurate.

In comparison, let us consider the procedure required to compute the sensitivities using the forward Euler implementation. Two strategies exist: First, if $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ in Eq. (2.17) is available, $\frac{d\boldsymbol{u}_\mathrm{f}}{d\boldsymbol{x}}$ is solved from the FE sensitivity analysis in Eqs. (2.16) and (2.17). Since $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ is not available *analytically*, one needs to compute $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ numerically, using a finite difference approach. This requires a complete mesh generation step for each perturbation. To compute $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$ accurately is computationally intensive due to the slow convergence rate. Alternatively, $\frac{d\boldsymbol{u}_\mathrm{f}}{d\boldsymbol{x}}$ can be computed directly via finite differences. The drawback here is that a complete mesh generation step and FE analysis are needed for each perturbation. Using this strategy, the increased computational cost due to the additional FE analyses is offset by using a coarser tolerance during mesh generation. From a computational cost perspective none of these strategies compare favourably with the analytical sensitivities available from Newton's method. However, if sensitivities are not required, as in gradient free optimization, the forward Euler method with a coarse tolerance is a feasible mesh

Table 2.4: Analytical and forward finite difference sensitivities calculated for the bow-tie structure depicted in Figure 2.4.

Point	Analytical ($\times 10^{-3}$)	Numerical ($\times 10^{-3}$)	Point	Analytical ($\times 10^{-3}$)	Numerical ($\times 10^{-3}$)
1	-0.111215	-0.111215	5	0.094632	0.094632
2	-1.788542	-1.788540	6	1.818842	1.818844
3	-0.043842	-0.043841	7	0.034656	0.034656
4	-0.002512	-0.002512	8	0.001624	0.001623

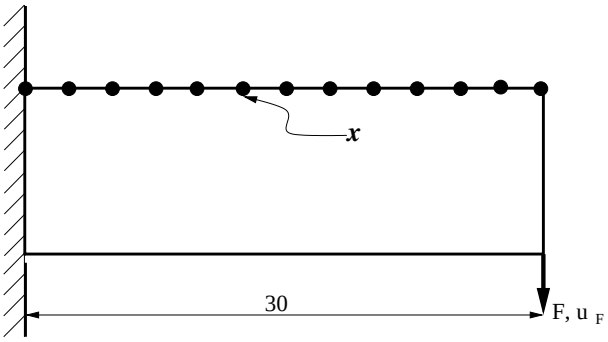


Figure 2.5: Initial structure and definition for the cantilever beam.

generation option in shape optimization.

2.4 Numerical examples

We implement two gradient based optimization algorithms, namely the Dynamic-Q method developed by Snyman and Hay [56], and the well known sequential quadratic programming (SQP) method [4]. All problems are allowed a maximum of 100 iterations, although in most cases the best objective function values f^{best} are obtained within 20 iterations for both algorithms. All the linear elastic FE analyses are performed using $E = 200 \times 10^3$ for Young's modulus, $\nu = 0.3$ for Poisson's ratio and thickness 1.0, under plane stress conditions. We investigate the effect of the number of control points (NCP).

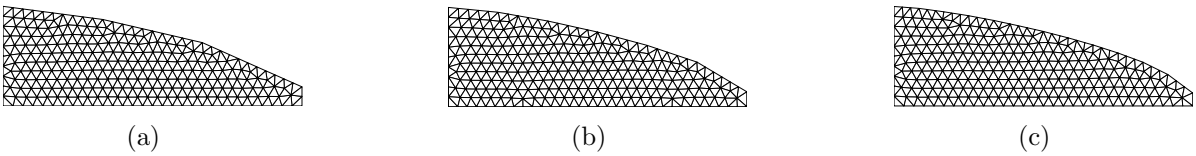
2.4.1 Example problem 1: Cantilever beam

Consider the cantilever beam depicted in Figure 2.5. The domain has a predefined length of 30 and a maximum allowable height of 10. The objective is to minimise u_{F}, the vertical displacement at the point of load application, subject to a maximum volume constraint of 70%. The magnitude of F is 10 N. The problem is conducted for an ideal element length of $h_0 = 1.0$. The control points are linearly spaced along the length of the top of the cantilever beam, as indicated in Figure 2.5.

The results obtained with both algorithms are summarized in Table 2.5. The optimal

Table 2.5: Best function value obtained for the cantilever beam problem.

NCP	Dynamic-Q	SQP
4	1.0073×10^{-2}	1.0068×10^{-2}
7	1.0011×10^{-2}	1.0013×10^{-2}
13	0.9996×10^{-2}	1.0005×10^{-2}



(a) (b) (c)

Figure 2.6: Cantilever beam: optimal shapes for a) 4, b) 7 and c) 13 control points respectively obtained with the Dynamic-Q algorithm.

shapes obtained with the Dynamic-Q algorithm are depicted in Figure 2.6; the SQP shapes are similar and therefore not shown. It is clear that as the NCP increases, the optimal designs converge.

2.4.2 Example problem 2: Full spanner

Consider the full spanner problem depicted in Figure 2.7. The structure has a predefined length of 24 and a maximum allowable height of 10. The control points are linearly spaced along the length of the spanner with half the control points describing the bottom and half the top of the geometry, as indicated in Figure 2.7. Note that no control points are used to describe the left- and right-most extremities of the spanner, which are stationary as indicated. The objective is to minimise $\frac{1}{2}(u_{\mathrm{FA}} - u_{\mathrm{FB}})$, with u_{FA} and u_{FB} the vertical displacement at the point of load application, for the two load cases $\mathrm{F_A}$ and $\mathrm{F_B}$ respectively. The corresponding boundary conditions for each load case are indicated by A and B respectively in Figure 2.7. In addition the problem is subjected to a maximum volume constraint of 30% of the defined domain together with a minimum handle thickness of 2. The loads $\mathrm{F_A}$ and $\mathrm{F_B}$ both have a magnitude of 10 N. The meshes are generated for an ideal element length h_0 of 0.5.

This problem should result in a symmetric geometry. However, symmetry is *not* enforced; deviations from symmetry are used to qualitatively evaluate the obtained designs.

The results obtained with both algorithms are summarized in Table 2.6. The optimal shapes obtained with the Dynamic-Q algorithm are depicted in Figure 2.8; again the SQP

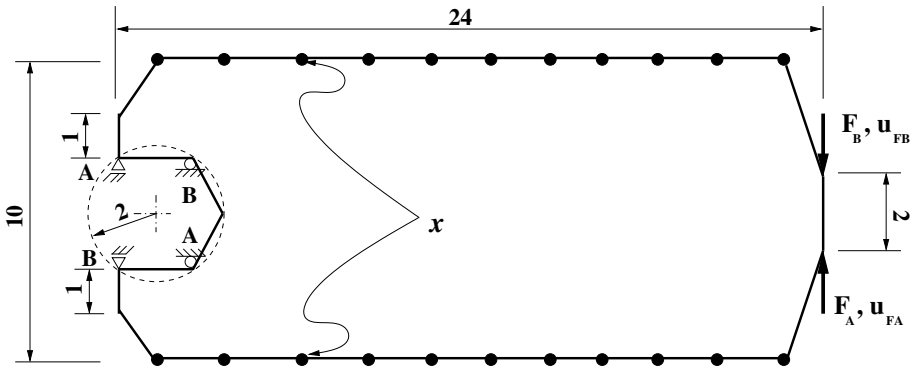
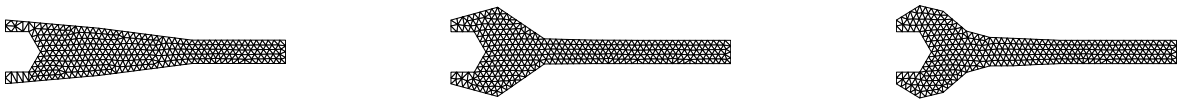


Figure 2.7: Initial structure and loads for the full spanner problem.

Table 2.6: Best function value obtained for the full spanner problem.

NCP	Dynamic-Q	SQP
4	5.7917×10^{-1}	5.7934×10^{-1}
10	3.5133×10^{-1}	3.5150×10^{-1}
22	3.1911×10^{-1}	3.1905×10^{-1}



(a) (b) (c)

Figure 2.8: Full spanner problem: optimal shapes for a) 4, b) 10 and c) 22 control points respectively obtained by the Dynamic-Q algorithm.

shapes are similar and therefore not shown. From Figure 2.8 it is evident that symmetric designs are obtained in all cases.

2.4.3 Example problem 3: Michell-like structure

The geometry [20] for this problem is depicted in Figure 2.9. The structure has a predefined length of 15 and a maximum allowable height of 10. The control points are linearly spaced along the length of the Michell-like structure with two additional control points describing the top as opposed to the bottom of the structure, as depicted in Figure 2.9. The objective is to minimise u_{F}, the vertical displacement at the point of load application, subject to a maximum volume constraint of 50%. The magnitude of F is 10 N. The problem is conducted for an ideal element length of $h_0 = 0.75$.

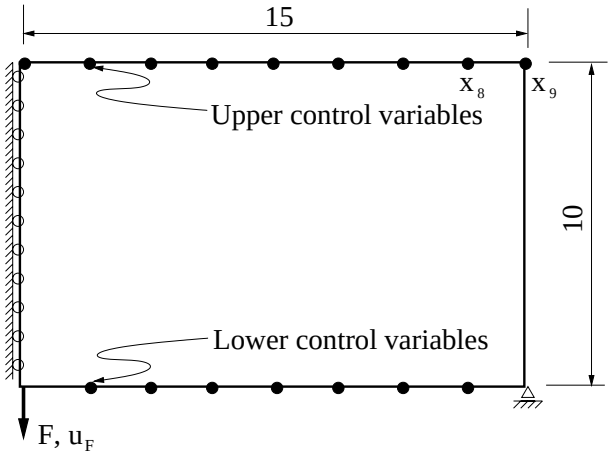
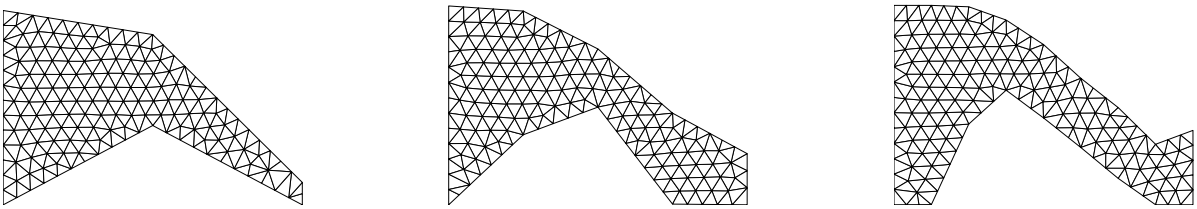


Figure 2.9: Initial structure and definition of half the Michell-like structure. (Control points x_8 and x_9 are indicated for the 16 NCP problems.)

Table 2.7: Best function value obtained for the Michell structure problem.

NCP	Dynamic-Q	SQP
4	1.4847×10^{-3}	1.4327×10^{-3}
8	1.2690×10^{-3}	1.2722×10^{-3}
16	1.2038×10^{-3}	1.2084×10^{-3}



(a) (b) (c)

Figure 2.10: Michell-like structure: optimal shapes for a) 4, b) 8 and c) 16 control points respectively obtained with the Dynamic-Q algorithm.

The results are summarized in Table 4.2 and Figure 2.10. As before, both algorithms obtain essentially the same solutions. One peculiarity however, is the optimal shape depicted in Figure 2.10(c). The shape of the right tip of the structure is counter intuitive; the origin thereof is the topic of the next section.

Objective function characteristics

Our resulting objective function contain numerically induced discontinuities since we use remeshing in our shape optimization strategy. Even with the presence of these discontinuities we were able to obtain intuitive designs for the cantilever beam and spanner problems. However, the Michell structure resulted in a counter intuitive design, which we expect to be a complication of these discontinuities in the objective function.

To investigate the nature of the objective function of the Michell structure, the two rightmost upper control variables x_8 and x_9 (see Figure 2.9) are perturbed. These control variables are varied between -1 and 1 (with equal intervals of 0.05), about the best objective function value found by the Dynamic-Q algorithm for the 16 NCP problem.

As shown in Figure 2.11, the objective function is discontinuous and local minima are present. These local minima and discontinuities are not physical phenomena but are purely due to remeshing. In fact, the objective function discontinuity is due to the mesh discontinuity, i.e. a change in a control variable value leads to the introduction or removal of an additional element. This is depicted in Figure 2.11: a small increase in x_9 results in 5 elements (top insert in Figure 2.11) instead of 4 elements (bottom insert in Figure 2.11) on the rightmost edge of the structure.

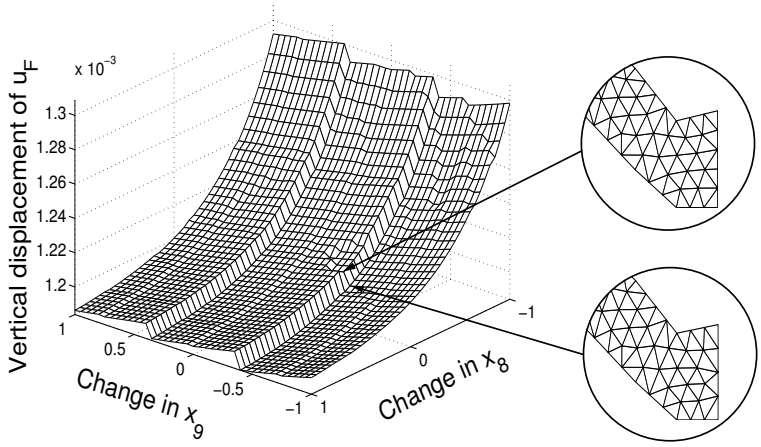
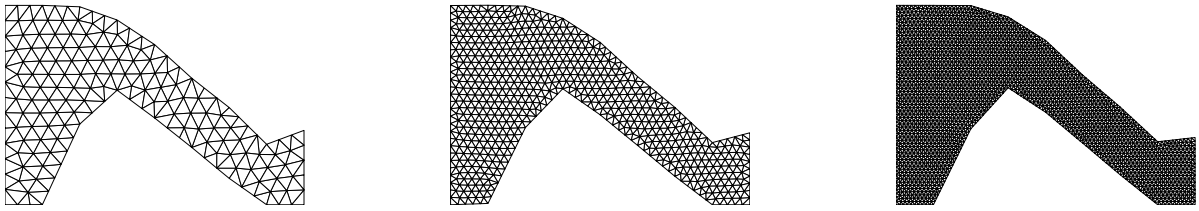


Figure 2.11: Vertical displacement at the point of load application for the variations of the 2 rightmost upper control variables (x_8, x_9) for the mesh depicted in Figure 2.10(c).



(a) (b) (c)

Figure 2.12: Michell-like structure: optimal shapes for ideal element lengths of a) 0.75, b) 0.375 and c) 0.1875 respectively for the 16 NCP problem obtained by the Dynamic-Q algorithm.

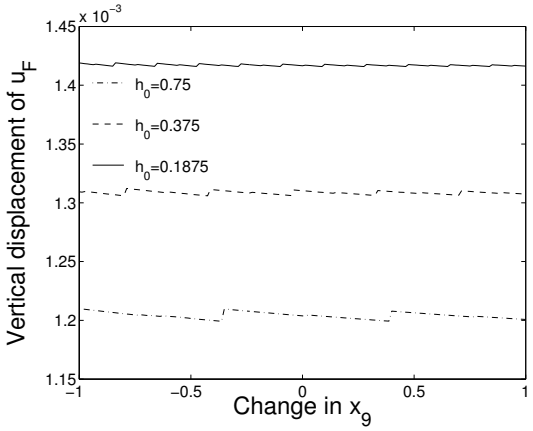
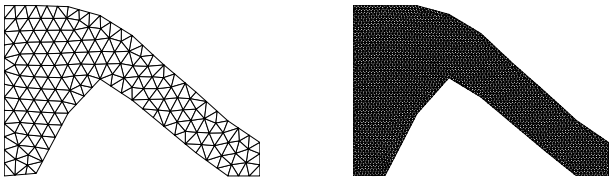


Figure 2.13: Vertical displacement at the point of load application for the variation of the rightmost upper control variable (x_9) for the meshes depicted in Figure 2.12(a), 2.12(b) and 2.12(c).



(a) (b)

Figure 2.14: Michell-like structure: effect of different starting points on the optimal shape for a) the course, and b) the fine mesh, obtained with the Dynamic-Q algorithm using 16 control points.

Since the mesh discontinuity is responsible for the discontinuities in Figure 2.11, it follows that a decrease in element size should decrease the magnitude of these discontinuities although their number will increase. To investigate the effect of h-refinement, the 16 NCP problem is repeated here for ideal element lengths of h_0 =0.375 and h_0 =0.1875. The optimal shapes are depicted in Figure 2.12. It is clear that as the element size decreases, the right tip of the structure gradually flattens off. To quantify the number of discontinuities and their magnitude, only the rightmost upper control variable x_9 is varied between -1 and 1 about the optimal designs depicted in Figure 2.12. Figure 2.13 confirms that as the element size decreases, the number of discontinuities increases, while the magnitudes decrease. These discontinuities are however still severe enough to adversely affect the performance of gradient based optimization algorithms.

To demonstrate that the geometric anomalies are indeed associated with local minima, we now restart the Dynamic-Q algorithm at an arbitrary starting point. (The results in the foregoing were all obtained for an initial volume fraction of 1.0.) For the coarse ($h_0 = 0.75$) and fine ($h_0 = 0.1875$) meshes respectively, the optimal shapes obtained are depicted in Figure 2.14(a) and 2.14(b).

For the coarse mesh, the objective function u_F decreases from 1.204×10^{-3} when

the geometric anomaly due to the local minimum is present, to 1.186×10^{-3}, which is (presumably) the global optimum. For the fine mesh, the objective function u_{F} decreases from 1.417×10^{-3} with the geometric anomaly due to the local minimum, to 1.395×10^{-3}, again presumably the global optimum.

While h-refinement does seem to reduce the severity of local minimum, h-refinement seems unable to assist in escaping from local minima. This observation is of course problem specific, and may even be proved incorrect in the limit of mesh refinement. However, for practical meshes, it is clear that solutions may be local minima when convex solvers are used. A simple multi-start strategy is likely to be of great benefit in attempting to find the global optimum (or at least some 'good' local optimum).

2.5 Conclusion

We have applied gradient based optimization techniques to shape design problems. In doing so, we have created a novel unstructured remeshing shape optimization environment, based on a truss structure analogy. The remeshing environment is quadratically convergent in solving for the equilibrium positions of the truss structure.

As expected the objective function value decreases as the number of control points are increased. This is a direct result of the number of possible design configurations that increases. However, due to the unstructured remeshing, discontinuities (local minima) are introduced into the optimization problem. In two of the three problems we studied, namely the cantilever and spanner design, these discontinuities did not seem to hamper the optimization. However, they hampered the optimization of the Michell structure as the final design converged to a counter intuitive design which we showed to be a local minimum caused by a discontinuity. Although the magnitude of these discontinuities decreases with mesh refinement, their number increases. For the gradient based algorithms, the severity of the anomaly is alleviated as the mesh is refined. Polynomial refinement e.g. linear strain triangles, may further decrease the magnitude of the discontinuities.

It is however suggested that local minima may efficient and effective be overcome with a simple multi-start strategy.

Even with the most inaccurate 2-D element available, namely the CST triangular element, we have demonstrated that gradient based algorithms are able to solve shape optimization problems efficiently using an unstructured remeshing strategy which makes analytical gradients available to the optimization algorithms.

CHAPTER 3

Applications of gradient-only optimization

In this chapter we study the minimization of numerically approximated objective functions containing non-physical jump discontinuities. These discontinuities arise when (partial) differential equations are discretized using non-constant methods and the resulting numerical solutions are used in computing the objective function. Although these functions may become discontinuous and non-differentiable we can compute exact gradient information where the function is differentiable and construct approximate gradient information where it is discontinuous. At a non-differentiable point, a partial derivative of the gradient vector is constructed by a one-sided directional derivative or given by the partial derivative itself, when the function is respectively non-differentiable or differentiable along the partial derivative direction. Such a constructed gradient field follows from the computational scheme, since every point has an associated discretization for which (semi) analytical sensitivities [40] of the numerically approximated optimization problem can be calculated. The only requirement is that we use a constant discretization topology when computing the sensitivities. Hence, from a computational perspective the gradient field of these discontinuous functions are defined everywhere albeit constructed at the discontinuities. In this study we refer to this gradient field as the *associated gradient* field. Rather than the construction of global approximations using only function value information to overcome the discontinuities, we propose to use only *associated* gradient information.

We elaborate on the modifications of classical gradient based optimization algorithms for use in gradient-only approaches, and we then present gradient-only optimization strategies using both BFGS and a new spherical quadratic approximation for sequential approximate optimization (SAO). We then use the BFGS and SAO algorithms to solve three problems of practical interest, both unconstrained

and constrained.

This chapter develops as follows: An overview of discontinuous objective functions in optimization is given in Section 3.1. We then present the classical mathematical programming problem and the corresponding gradient-only optimization problem in Section 3.3. The optimization algorithms used in this study are then outlined in Section 3.4. We then consider three example problems in the remainder of Section 3.5. These are a one dimensional transient heat transfer problem; an *unconstrained* and a *constrained* shape design problem, and lastly a material identification study. Finally, we offer conclusions in Section 3.6. The derivation of the required sensitivities for the test problems is outlined in an Appendix.

3.1 Introduction

Many problems in engineering and the applied sciences are described by ordinary or partial differential equations (ODEs/PDEs), e.g. the well-known elliptical PDEs of structural mechanics. Analytical solutions to these are seldom available and in many cases, (approximate) numerical solutions need to be computed.

Temporal (P)DEs may be solved using fixed or variable time steps; for spatial (P)DEs, the equivalents are fixed and mesh moving spatial updating strategies on the one hand, and remeshing on the other. Fixed time steps and mesh moving strategies however may imply serious difficulties, e.g. impaired convergence rates and highly distorted grids and meshes, which may even result in failure of the computational procedures used. The variable or 'non-constant' strategies are preferable by far.

(P)DEs also often describe the physics of some problem that is to be optimized using numerical optimization techniques. Indeed, the numerical solution of (P)DE problems is regularly used to numerically approximate optimization problems in science and engineering. However, variable methods now become problematic since they may result in non-smooth or step-discontinuous objective functions of the design variables, whereas the 'constant strategies' result in smooth continuous objective functions. The step discontinuities resulting from the variable methods are non-physical, since they are mere artifacts of the numerical strategies used to approximate the inherently smooth objective function of the exact optimization problem described by the (P)DE under consideration. Although these functions may become discontinuous and non-differentiable we can compute exact gradient information where the function is differentiable and construct approximate gradient information where it is discontinuous. At a non-differentiable point, a partial derivative of the gradient vector is constructed by a one-sided directional derivative or given by the partial derivative itself, when the function is respectively non-differentiable or differentiable along the partial derivative direction. Such a constructed gradient field follows from the computational scheme, since every point has an associated discretization for which (semi) analytical sensitivities [40] of the numerically approximated optimization problem can be calculated. The only requirement is that we use a constant discretization topology when computing the sensitivities. Hence, from a computational perspective the gradient field of these discontinuous functions are defined everywhere albeit constructed at the discontinuities. In this study we refer to this gradient approximation as the *associated gradient*, which we denote $\boldsymbol{\nabla}_A f(\boldsymbol{x})$ which gives an *associated directional derivative* when $\boldsymbol{\nabla}_A f(\boldsymbol{x})$ is used to compute the directional derivative.

Consider Figure 3.1, which depicts three functions that describe an optimization problem. $f_A(x)$ is the unknown analytical function to an exact optimization problem described by a system of partial differential equations; $f_N(x)$ is the numerically computed piece-wise smooth step discontinuous objective function which is an approximation to $f_A(x)$; $f_C(x)$

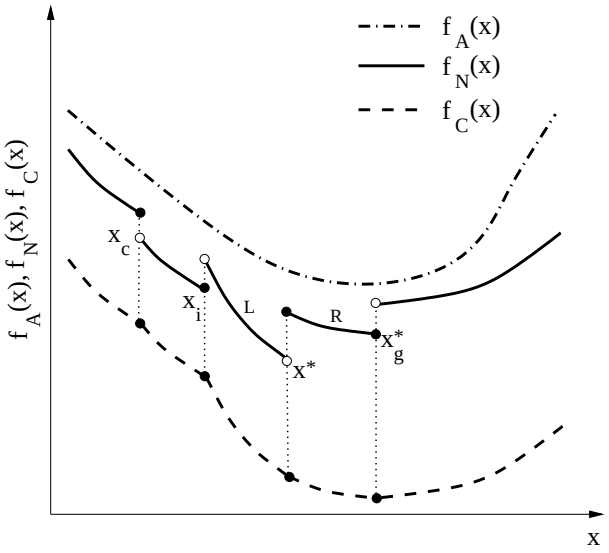


Figure 3.1: Plot depicting a piece-wise smooth step discontinuous numerical objective function $f_N(x)$ together with the corresponding underlying (unknown) continuous analytical objective function $f_A(x)$ of an optimization problem. In addition we depict a projected piece-wise smooth continuous objective function $f_C(x)$ obtained by removing the step discontinuities from $f_N(x)$.

is a constructed piece-wise smooth continuous objective function from $f_N(x)$ by removing the discontinuities. The optimum x^* and positive associated gradient projection point x_g^*, of $f_N(x)$ are also indicated. We refer to x_g^* as a positive associated gradient projection point since the *associated directional derivatives* in all directions around that point are positive. However, to avoid using long descriptive names we will merely refer to x_g^* as a positive projection point. The discontinuities in $f_N(x)$ as well as the *associated gradient* field (and *associated directional derivatives*) are a direct consequence of a sudden change in the discretization used to compute an approximate solution to a system of partial differential equations.

As long as the changes in the discretization vary smoothly, the underlying objective function is smooth. This is obtained when using constant discretization strategies, since the underlying discretization errors vary smoothly. The use of such smooth objective functions are prevalent in engineering optimization [34, 45]. When the objective function is smooth, as opposed to the objective function depicted in Figure 3.1, the optimum x^* and positive projection point x_g^* define the same point. This point is usually assumed to be close to the exact optimizer, in particular when good numerical strategies are used.

If however when the discretization changes abruptly, a step discontinuity results. The reponse could be suddenly underestimated (as depicted by x_c), or overestimated (as depicted by x_i), as demonstrated in Figure 3.1. We refer to these two points respectively as a consistent (x_c) and inconsistent (x_i) step discontinuities. The function and the slope of the function (*associated derivative*) around the consistent discontinuity indicates descent. Around an inconsistent discontinuity the function value indicates ascent, as

opposed to descent by the slope, and are hence inconsistent with each other. Therefore, inconsistent discontinuities would be completely ignored when conducting optimization that only relies on the *associated derivative*, whereas they would manifest as local minima when conducting optimization that also uses function values.

The premise of this study is that the piece-wise continuous parts in the numerically approximated objective function $f_N(x)$, where the error is smoothly varying, better describes the behaviour of the underlying unknown analytical function $f_A(x)$ than the abrupt changes in the discretization error. This premise holds in particular, when rediscretization is done to increase the accuracy of the analysis by reducing the discretization error. The motivation for our premise is as follows: consider the optimum x^* that occurs over an inconsistent discontinuity, as depicted in Figure 3.1. The *associated derivative* at x^* indicates further possible improvement if x is increased. It is reasonable to believe that had the discretization been varied smoothly with no discontinuity present, then the function value would continue to decrease as the error would change smoothly as before. However, when rediscretization is conducted the discretization error changes abruptly and hence introduces a discontinuity. The function value and *associated gradient* vector obtained after the rediscretization is more accurate if we assume that the rediscretization increases the accuracy of the analysis and consequently reduces the discretization error. If the function continues to decrease along this direction (even if it starts at a higher value), then the optimum at the discontinuity could be safely discarded as an unwanted or inferior solution to the optimization problem. A more suitable solution to the optimization problem would then be a local minimizer of the piece-wise smooth *continuous* function $f_C(x)$, as depicted in Figure 3.1. However, such a point is also characterized by a positive projection point x_g^* of $f_N(x)$, as depicted in Figure 3.1.

Consider the positive projection point x_g^* that occurs over a discontinuity as depicted by $f_N(x)$ in Figure 3.1, with a piece-wise smooth part L of the function to the left and a piece-wise smooth part R of the function to the right of it. If rediscretization was omitted and the piece-wise smooth part L extended an optimal solution would occur to the right of x_g^* and similarly an optimum to the left by the extended piece-wise smooth part R. Therefore, x_g^* would be bounded by the two optima of the two extended piece-wise continuous pieces and consequently within a domain of uncertainty of the numerical model. In order to distinguish between these three possible solutions one would have to increase the accuracy of the numerical model or obtain information regarding the absolute error w.r.t the exact solution, which is usually not available. Lastly, when the optimum $\boldsymbol{x}^*$ and the positive projection point $\boldsymbol{x}_g^*$ coincide the use of only the *associated derivative* is an efficient strategy to ignore the local minima introduced by inconsistent step discontinuities.

Hence, variable time step methods and variable remeshing techniques are normally avoided in optimization, due to the very fact that the discontinuities present may cause

numerous difficulties during the optimization steps. An important spatial example is structural shape optimization, in which fixed or moving grid strategies are almost always used; the very motivation for this being that remeshing strategies cannot be used efficiently in optimization, due to the induced discontinuities during optimization, e.g. see References [1, 9, 32, 39].

The reasons for this are obvious: discontinuous cost functions are difficult to optimize in comparison to smooth convex cost functions, since the discontinuities may introduce spurious or false local minima which may make the use of efficient gradient based optimization algorithms difficult, if not impossible. Accordingly, the optimization of discontinuous functions usually requires highly specialized optimization strategies, and possibly, heuristic approaches.

For the continuous programming problem, many well known minimization algorithms are available, e.g. steepest descent, (preconditioned) conjugate gradient methods, and variable metric methods like BFGS. However, if f is discontinuous, i.e. $f \notin C^0$, the minimizer $\boldsymbol{x}^*$ of the mathematical programming problem may not satisfy the standard optimality criteria. Indeed, the optimality criteria may not even be defined. Accordingly, the well-known efficient optimization algorithms mentioned above may be unable to minimize the resulting step discontinuous function. Conventional gradient based optimization approaches have been used in restart strategies that restarts the optimization process after a discontinuity to continue with the conventional optimization process [26, 30]. Such an approach would eventually converge to a positive projection point x_g^* and not necessarily the optimum x^* as might be expected. The computational efficiency of such an approach however requires two analysis per design over a discontinuity in addition to the computations required to identify the discontinuity.

As a solution strategy, some researchers in structural optimization have resorted to surrogate optimization in which approximations are constructed using *function values only*, in combination with design-of-experiments (DoE) techniques. The resulting approximations are smooth, and are often assumed to be valid over large regions of the design domain. Known as so-called *global approximations*, they may be optimized using gradient based techniques, if so desired. (Although the use of function values only is popular, gradient information is sometimes also used. In addition, 'mid-range' surrogates are sometimes also constructed, in combination with elaborate strategies for controlling the step sizes, and the acceptance and rejection of points.) A drawback of surrogate optimization is that problems are limited to moderate dimensions since the algorithms scale poorly with dimensionality. For an overview of these methods, see Barthelemy and Haftka [3], Haftka and Gurdal [24], Sacks *et al.* [47], Toropov [59], and many others. The interested reader is also referred to the recent review paper by Simpson *et al.* [51] as well as the paper by Forrester and Keane [19].

Accurate *associated gradient* information of the numerically approximated optimiza-

tion problem is indeed available for the (P)DEs that occur so frequently in engineering and applied mathematics. This does of course not hold for 'inherently noisy objective functions' like crash-simulations, etc. An important example of non-physical discontinuities introduced by variable discretization methods occurs in shape optimization, a problem we will elaborate on in some detail herein.

Yet, few gradient-only optimization algorithms seem to be available, even though gradient-only optimization algorithms are by no means new to optimization. The first gradient-only optimization algorithm developed is the widely known Newton's method, e.g. see Wallis [64]. However, Newton's method has several drawbacks for practical optimization. Firstly, it requires that a function be twice-differentiable, since Hessian information is needed to compute the update steps. Secondly, Newton's method locates diminishing gradients which may converge to suboptimal designs. In addition, Newton's method may oscillate around an optimal solution or even diverge.

To the best of our knowledge, only a few other gradient-only optimization algorithms have been developed, see References [4, 43, 44, 49, 53, 54]. Most of these algorithms also address some of the major difficulties of Newton's method.

Gradient-only optimization has also been extended to nonsmooth functions with the introduction of subdifferentials[1] by Shor *et al.* [49]; these methods are used to compute and define the gradient of a function where it is not differentiable. Subdifferentials are however defined as the set of subgradients for which a hyperplane (linearization of the function) defined by the subgradient at the discontinuity is less or equal to the actual function value. At a step or jump discontinuity the subdifferential is the empty set at the side of the discontinuity where the hyperplane lies above the function. In addition subgradient methods reduces to steepest descent methods with *a priori* selected step lengths which often require tuning (in particular for larger problems [5]) and Lipschitz conditions regarding the function. Although the use of *a priori* selected step lengths increases the flexibility of the algorithm (e.g. when optimizing piece-wise linear functions) it increases the computational cost when the additional flexibility is not required. Lastly, all these methods are employed and designed to find the point for which the objective function is a minimum, with the exception of Newton's method which is often employed to merely find stationary points of a function.

Our gradient-only approach however solely aims to find positive projection points $\boldsymbol{x}_g^*$. For smooth convex functions this is equivalent to finding the optimum $\boldsymbol{x}^*$ of the function. However, when piece-wise smooth step discontinuous functions are considered, the optimum $\boldsymbol{x}^*$ may be distinct from the positive projection point $\boldsymbol{x}_g^*$ as they define two generalizations of solutions for piece-wise smooth step discontinuous functions. Our gradient-only approach merely requires that the gradient field be everywhere computable as for example with *associated gradients*. Gradient-only optimization approaches are

[1] A subdifferential is a set of subgradients at a point.

merely conventional gradient based optimization algorithms slightly modified to use only gradient information, and have comparable computational efficiency to conventional gradient based optimization algorithms. In addition, we require no assumptions regarding the Lipschitz conditions of a function. Consider again Figure 3.1 for which we note that by only considering the *associated derivative* of $f_N(x)$ during optimization we effectively optimize $f_C(x)$ without having to go through the computational effort of distilling $f_C(x)$ from $f_N(x)$, since the *associated derivative* field of $f_N(x)$ and $f_C(x)$ are the same. Gradient-only optimization effectively reduces the complexity of functions plagued with numerically induced step or jump discontinuities by acting as a filter to ignore these discontinuities, on the condition that a positive projection point (defined as the solution to the gradient-only optimization problem presented herein) is a suitable solution for the problem at hand.

If it is known or assumed that $\boldsymbol{x}_g^*$ coincides with a minimizer $\boldsymbol{x}^*$ of $f(\boldsymbol{x})$ then these problems can be approached from a classical mathematical programming perspective which have to be solved using global optimization algorithms. This is due to the numerically induced step discontinuities that manifest as local minima in the function domain. However, the resulting discontinuous problems may still be optimized efficiently using gradient-only optimization as the numerically induced step discontinuities are ignored, which is an alternative to constructing global approximations.

Lastly, both the function values and analytically computed *associated derivatives* are consistently computed for a numerically computed objective function and prone to discretization errors, in particular discretization errors that changes abruptly. We do however note that inconsistent step discontinuities in the function value may result in local minima and trap conventional optimization approaches. However, step discontinuities in the *associated derivative* from our experience mostly result in abrupt changes in the magnitude of the *associated derivative* but not the sign of the *associated derivative* which indicates the direction of ascend of $f(x)$. This only affects the convergence rate of the gradient-only optimization approaches and not the robustness thereof. If however, the error in the *associated derivative* changes the sign of the *associated derivative*, it would manifest as a positive projection point. However, such a point would also manifest as a local minimum in the function value. We show numerically that our premise and gradient-only approaches yield promising results for multidimensional problems.

To ease the presentation of the remainder of this chapter we will merely refer to f as the objective function for which we would like to find the *positive projection point* $\boldsymbol{x}_g^*$, but this implies the numerically approximated objective function f_N.

3.2 Definitions

The functions we consider in this study are step discontinuous and therefore not everywhere differentiable. However computationally the derivatives and gradients are everywhere computable since the analysis is per se restricted to the part of the objective function before, or after a discontinuity. We therefore define an *associated derivative* $f'^{A}(x)$ and *associated gradient* $\nabla_A f(\boldsymbol{x})$ which follow computationally when the sensitivity analysis is consistent [48]. Firstly, we define the *associated derivative*

Definition 3.2.1. Let $f : X \subset \mathbb{R} \to \mathbb{R}$ be a real *univariate* piece-wise smooth step discontinuous function that is everywhere defined. The *associated derivative* $f'^{A}(x)$ for $f(x)$ at a point x is given by the derivative of $f(x)$ at x when $f(x)$ is differentiable at x. The *associated derivative* f'^{A} for $f(x)$ non-differentiable at x, is given by the left-sided derivative of $f(x)$ when x is associated to the left piece-wise continuous section of the discontinuity, otherwise it is given by the right-sided derivative.

Secondly, the *associated gradient* is defined as follows:

Definition 3.2.2. Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real *multivariate* piece-wise smooth step discontinuous function that is everywhere defined. The *associated gradient* $\nabla_A f(\boldsymbol{x})$ for $f(\boldsymbol{x})$ at a point $\boldsymbol{x}$ is given by the gradient of $f(\boldsymbol{x})$ at $\boldsymbol{x}$ when $f(\boldsymbol{x})$ is differentiable at $\boldsymbol{x}$. The *associated gradient* $\nabla_A f(\boldsymbol{x})$ if $f(\boldsymbol{x})$ is non-differentiable at $\boldsymbol{x}$ is defined as the vector of partial derivatives where each partial derivative is an *associated derivative* (see Definition 4.2.1).

It follows from Definitions 4.2.1 and 4.2.2 that the *associated gradient* reduces to the gradient of a function that is everywhere differentiable.

Secondly, we present definitions for univariate and multivariate associated gradient unimodality based solely on the associated gradient field of a real valued function [4].

Definition 3.2.3. A univariate function $f : X \subset \mathbb{R} \to \mathbb{R}$ with associated derivative $f'^{A}(\lambda)$ uniquely defined for every $\lambda \in X$, is (resp., strictly) associated derivative unimodal over X if there exists a $x_g^* \in X$ such that

$$f'^{A}(x_g^* + \lambda u)u \geq (\text{resp., } >)\ 0,\ \forall\ \lambda \in \{\beta : \beta > 0 \text{ and } \beta \subset \mathbb{R}\} \\ \text{and}\ \ \forall\ u \in \{-1, 1\} \text{ such that } [x_g^* + \lambda u] \in X. \quad (3.1)$$

We now consider (resp., strictly) associated derivative unimodality for multivariate functions [46].

Definition 3.2.4. A multivariate function $f : X \subset \mathbb{R}^n \to \mathbb{R}$ is (resp., strictly) associated derivative unimodal over X if for all $\boldsymbol{x}^1$ and $\boldsymbol{x}^2 \in X$ and $\boldsymbol{x}^1 \neq \boldsymbol{x}^2$, every corresponding

univariate function

$$F(\lambda) = f(\boldsymbol{x}^1 + \lambda(\boldsymbol{x}^2 - \boldsymbol{x}^1)), \quad \lambda \in [0, 1] \subset \mathbb{R}$$

is (resp., strictly) associated derivative unimodal according to Definition 4.4.2.

Note that our definition of associated derivative unimodality excludes functions that are unbounded in the associated derivative although such functions are bounded from below e.g. $f(x) = -\frac{x}{2} + \lfloor |x| \rfloor$.

3.3 Problem formulation

We consider both *unconstrained* and *constrained* optimization problems; we depart with the former.

3.3.1 Unconstrained minimization problem

Consider the following general unconstrained minimization problem:

Formulation 3.3.1. *Let $f(\boldsymbol{x}^*)$ be a real-valued function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ that is bounded from below. Find the minimum value $f(\boldsymbol{x}^*)$, such that*

$$f^* = f(\boldsymbol{x}^*) \ \{<\} \leq \ f(\boldsymbol{x}), \ \forall \ \boldsymbol{x} \in X, \tag{3.2}$$

with X the convex set of all possible solutions. □

If the function f is unimodal and at least twice continuously differentiable, i.e. $f \in C^2$, the minimizer $\boldsymbol{x}^* \in X$ is characterized by the optimality criterion

$$\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{0}, \tag{3.3}$$

with $\boldsymbol{H}(\boldsymbol{x}^*)$ semi-positive definite. Here, $\boldsymbol{\nabla}$ represents the gradient operator, and $\boldsymbol{H}$ the Hessian matrix.

However, if f is discontinuous, i.e. $f \notin C^0$, the minimizer $\boldsymbol{x}^*$ of the mathematical optimization problem (3.2) may not satisfy the optimality criterion given in (3.3). Indeed, the optimality criterion (3.3) may not even be defined. If f is an associated derivative unimodal discontinuous function with *associated gradient* field everywhere defined, an alternative generalization to Formulation 3.3.1 may be written in derivative form as follows:

Formulation 3.3.2. *Find the non-negative gradient projection point* $\boldsymbol{x}_g^*$ *(hereafter referred to as a positive projection point), of a given* {*strictly*} *associated derivative unimodal real-valued function* $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$*, such that*

$$\nabla_A^T f(\boldsymbol{x}^* + \delta \boldsymbol{u})\boldsymbol{u} \; \{>\} \geq \; 0, \; \forall \; \boldsymbol{u} \in \mathbb{R}^n \; \textit{and} \; \forall \; (\delta > 0) \in \mathbb{R} \; \textit{such that} \; \boldsymbol{x}^* + \delta \boldsymbol{u} \in X, \quad (3.4)$$

with X *the convex set of all possible solutions.* □

Formulation 3.3.2 implies, in the strict case, that departure from the positive projection point $\boldsymbol{x}_g^*$ in any search direction $\boldsymbol{u} \in \mathbb{R}^n$, and for any step size $\delta > 0$, results in positive *associated directional derivatives*. It follows that the sign of the projected gradient onto the search direction along some descent direction $\boldsymbol{u}$ changes from negative to positive at the positive projection point $\boldsymbol{x}^*$.

For f smooth, i.e. $f \in C^2$, Formulations 3.3.1 and 3.3.2 are equivalent, since the condition $\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{0}$ follows, and the positive projection point $\boldsymbol{x}^*$ in Formulation 3.3.2 is identical to the minimizer $\boldsymbol{x}^*$ in Formulation 3.3.1. The second order condition i.e. $\boldsymbol{H}$ positive definite is implied by the requirement that the *associated directional derivatives* w.r.t. all search directions $\boldsymbol{u}$ is larger than 0. For associated derivative unimodal step discontinuous objective functions, Formulations 3.3.1 and 3.3.2 may define different solutions.

3.3.2 Equality constrained minimization problems

Next, we consider the following general equality constrained minimization problems:

Formulation 3.3.3. *Find the minimum value* $f(\boldsymbol{x}^*)$ *of a given real-valued function* $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$*, such that*

$$f^* = f(\boldsymbol{x}^*) \; \{<\} \leq \; f(\boldsymbol{x}), \; \forall \; \boldsymbol{x} \in X, \; \textit{such that} \; h_j(\boldsymbol{x}) = 0, j = 1, 2, \ldots, r \leq n, \quad (3.5)$$

with X *the convex set of all possible solutions and with* $h_j : X \subseteq \mathbb{R}^n \to \mathbb{R}, j = 1, 2, \ldots, r \leq n$. □

For smooth objective and constraint functions, we can transform problem (3.5) into an unconstrained optimization problem via the Lagrangian function

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{j=1}^{r} \lambda_j h_j(\boldsymbol{x}), \quad (3.6)$$

which allows us to solve (3.5) using the dual formulation

$$\max_{\boldsymbol{\lambda}} \{ \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) \}. \quad (3.7)$$

If the objective and constraint functions are convex then the solution of Eq. (3.7) is also the optimum.

Finally, we progress to discontinuous equality constrained optimization problems that is smooth around the positive projection point $\boldsymbol{x}_g^*$ in the objective and constraint functions:

Formulation 3.3.4. *Find the positive projection point $\boldsymbol{x}_g^*$ of a given {strictly} associated derivative unimodal real-valued function $f : X \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ that, such that*

$$\begin{aligned} \nabla_A^T f(\boldsymbol{x}_g^* + \delta\boldsymbol{u})\boldsymbol{u} \; \{>\} \geq \; 0, \; \forall \; \boldsymbol{x}_g^* + \delta\boldsymbol{u} \in X, \\ \textit{such that } h_j(\boldsymbol{x}_g^*) = 0, \;\; \forall \; j = 1, 2, \ldots, r \leq n, \\ \textit{and } \boldsymbol{u} \in \{\bar{\boldsymbol{u}} : \nabla_A^T h_j(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}} = 0, \;\; \forall \; j = 1, 2, \ldots, r \leq n, \} \end{aligned} \tag{3.8}$$

with X the convex set of all possible solutions and δ a real positive value and $\boldsymbol{x}_g^* \in X$. □

The condition $\nabla_A^{\mathrm{T}} h_j(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}} = 0$ [46], reduces the set of projection directions $\boldsymbol{u}$ to a set of feasible directions. Firstly, we construct the Taylor expansion of the j^{th} equality constraint $h_j(\boldsymbol{x}_g^*)$ around $\boldsymbol{x}_g^*$ to give

$$\tilde{h}_j(\boldsymbol{x}_g^* + \bar{\boldsymbol{u}}) = h_j(\boldsymbol{x}_g^*) + \nabla_A^{\mathrm{T}} h_j(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}} + \mathcal{O}\left((\bar{\boldsymbol{u}})^2\right), \tag{3.9}$$

which reduces to

$$h_j(\boldsymbol{x}_g^* + \bar{\boldsymbol{u}}) \approx h_j(\boldsymbol{x}_g^*) + \nabla_A^{\mathrm{T}} h_j(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}}. \tag{3.10}$$

Since $\boldsymbol{x}_g^*$ is feasible by definition we have $h_j(\boldsymbol{x}_g^*) = 0$, in addition we require Eq. (3.10) to be feasible which gives

$$h_j(\boldsymbol{x}_g^* + \bar{\boldsymbol{u}}) \approx h_j(\boldsymbol{x}_g^*) + \nabla_A^{\mathrm{T}} h_i(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}} = \nabla_A^{\mathrm{T}} h_j(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}} = 0, \tag{3.11}$$

and similarly Eq. 3.11 needs to hold for all $j = 1, 2, \ldots, n \leq r$ constraints. Consequently, $\boldsymbol{u}$ is required be in the set $\{\bar{\boldsymbol{u}} : \nabla_A^{\mathrm{T}} h_j(\boldsymbol{x}_g^*)\bar{\boldsymbol{u}} = 0, \;\; \forall \; j = 1, 2, \ldots, r \leq n, \}$.

Again using the Lagrangian function we may transform problem (3.8) into an unconstrained optimization problem.

This time in solving (3.8) we use a gradient-only dual formulation

$$\max_{\boldsymbol{\lambda}}^{g}\{\min_{\boldsymbol{x}}^{g} L(\boldsymbol{x}, \boldsymbol{\lambda})\}, \tag{3.12}$$

with $\max_{\boldsymbol{\lambda}}^{g}$ defined as follow: Find $\boldsymbol{\lambda}$, such that

$$\nabla_{A_{\boldsymbol{\lambda}}}^{\mathrm{T}} L(\boldsymbol{x}, \boldsymbol{\lambda} + \gamma_v \boldsymbol{v})\boldsymbol{v} \leq 0, \; \forall \; \boldsymbol{v} \in \mathbb{R}^r, \tag{3.13}$$

and similarly for $\min_{\boldsymbol{x}}^{g}$: Find $\boldsymbol{x}$, such that

$$\nabla_{A_{\boldsymbol{x}}}^{\mathrm{T}} L(\boldsymbol{x}+\delta_u \boldsymbol{u}, \boldsymbol{\lambda})\boldsymbol{u} \geq 0, \forall\ \boldsymbol{u} \in \mathbb{R}^n \text{ such that } \boldsymbol{x}+\delta_u \boldsymbol{u} \in X, \tag{3.14}$$

with X the convex set of all possible solutions, $\nabla_{A_{\boldsymbol{x}}}$ the *partial associated derivatives* w.r.t. $\boldsymbol{x}$, $\nabla_{A_{\boldsymbol{\lambda}}}$ the *partial associated derivatives* w.r.t. $\boldsymbol{\lambda}$ and δ_u and γ_v real positive numbers.

For step discontinuous functions, inconsistencies between Formulations 3.3.1 and 3.3.2 on the one hand, and Formulations 3.3.3 and 3.3.4 on the other may arise.

3.4 Optimization algorithms

Most (if not all) classical optimization algorithms can be modified to use only gradient information instead of both function value and gradient information. To illustrate, we consider two classes of algorithms, namely line search descent methods and approximation methods.

In classical line search descent methods, the search direction is obtained from gradient information. The step length is usually obtained using some line search strategy that uses function information. However, gradient-only line search descent methods may be formulated by merely changing the line search strategies to use only *associated gradient* information. In this study, we will consider a bracketing-interval line search strategy, although many alternative and more efficient line search strategies are available. (As an alternative, Snyman [54] presents a gradient-only implementation of an interpolation line search strategy.) In particular, we will consider the Broyden-Fletcher-Goldfarb-Shanno (BFGS) second order line search descent algorithm [55], which is suitable for problems that range from from small to large dimensionality. For large-scale problems [21], the limited memory BFGS algorithm [36] is indeed widely used. Lastly, the BFGS algorithm can be used without a line search strategy as a monotonically decreasing superlinear convergent BFGS has been demonstrated using only fixed step length updates [68], when certain Lipschitz continuity assumptions regarding the objective function hold.

In sequential approximation optimization (SAO), the approximation functions are normally constructed using both function and gradient information. Function values may for example be used to approximate the curvature used in the approximations, as is done in spherical quadratic approximations, e.g. see Reference [52]. Again, gradient-only optimization is however possible.

We would like to emphasize that our choice for BFGS and SSA algorithms is arbitrary and merely to illustrate the ease with which we can transform conventional gradient based algorithms to be gradient-only optimization algorithms. We deliberately chose one line search algorithm and one approximation algorithm, as they represent two widely used classes of optimization algorithms. By following the same methodology other optimiza-

tion algorithms which require a decision to be made between a current and proposed solution using function value information, can be extended using the proposed methodology. Hence, to the best of our knowledge all conventional gradient based algorithms could be modified using our methodology. Considering the extension of gradient-only optimization to alternative classes of optimization algorithms, as for example population based methods, may prove a bit more challenging, in particular when the features of these methods are to be preserved.

We note that gradient-only algorithms based on classical optimization algorithms that scale well, may also be expected to scale well (provided the *associated gradient* computation scales well). Finite difference strategies may become prohibitively expensive to compute the full associated gradient vector for large-scale problems, but may be adequate for the computation of the *associated directional derivatives*[2]. Automatic differentiation or (semi-) analytical sensitivity analyses are alternatives to compute computationally efficient *associated gradient* information.

3.4.1 BFGS second order line search descent method

In the BFGS algorithm, the iterates are updated using

$$\boldsymbol{x}^{\{k\}} = \boldsymbol{x}^{\{k-1\}} + \lambda^{\{k\}}\boldsymbol{u}^{\{k\}}, \tag{3.15}$$

where the superscript k indicates the iteration number, $\boldsymbol{x}$ the design vector and $\boldsymbol{u}$ the descent direction. λ is a scalar value obtained from a line search.

The descent direction $\boldsymbol{u}^{\{k\}}$ is obtained from

$$\boldsymbol{u}^{\{k\}} = -\boldsymbol{G}^{\{k-1\}}\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}}), \tag{3.16}$$

and $\boldsymbol{G}^{\{k\}}$ is updated using

$$\boldsymbol{G}^{\{k\}} = \boldsymbol{G}^{\{k-1\}} + \left[1 + \frac{(\boldsymbol{y}^{\{k\}})^{\mathrm{T}}\boldsymbol{G}^{\{k-1\}}\boldsymbol{y}^{\{k\}}}{(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}\boldsymbol{y}^{\{k\}}}\right]\left[\frac{\boldsymbol{v}^{\{k\}}(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}}{(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}\boldsymbol{y}^{\{k\}}}\right] - \left[\frac{\boldsymbol{v}^{\{k\}}(\boldsymbol{y}^{\{k\}})^{\mathrm{T}}\boldsymbol{G}^{\{k-1\}} + \boldsymbol{G}^{\{k-1\}}\boldsymbol{y}^{\{k\}}(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}}{(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}\boldsymbol{y}^{\{k\}}}\right], \tag{3.17}$$

with

$$\boldsymbol{v}^{\{k\}} = \lambda^{\{k\}}\boldsymbol{u}^{\{k\}},$$

[2]Computation of accurate and consistent gradients using finite differences requires a temporary deactivation of the non-constant discretization strategy, in order to avoid a finite difference step over a discontinuity [48].

and

$$\boldsymbol{y}^{\{k\}} = \left(\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}})\right).$$

$\boldsymbol{G}^{\{0\}}$ is commonly initiated with the $n \times n$ identity matrix $\boldsymbol{I}$, and reset to $\boldsymbol{G}^{\{k\}} = \boldsymbol{I}$ after every n iterations [55].

Let us return to the line search strategies needed to determine the $\lambda^{\{k\}}$. The line search conducted at the k^{th} iteration from the point $\boldsymbol{x}^{\{k-1\}}$ along a descent direction $\boldsymbol{u}^{\{k\}}$ is described by the univariate function

$$F(\lambda) = f(\boldsymbol{x}^{\{k-1\}} + \lambda \boldsymbol{u}^{\{k\}}), \quad \lambda \geq 0 \subset \mathbb{R}. \tag{3.18}$$

Generally, line searches use function values to locate the minimum of $F(\lambda)$. Various strategies exist to conduct function value based line searches, e.g. explicit searches using golden section or implicit searches using some interpolation strategy with Powell's method, to name one example.

Alternatively, line searches can be conducted solely based on the *associated derivative* of $F(\lambda)$, with $F'^{A}(\lambda)$ given by

$$F'^{A}(\lambda) = \left[\nabla_A^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}} + \lambda \boldsymbol{u}^{\{k\}})\right] \boldsymbol{u}^{\{k\}}. \tag{3.19}$$

Line searches using function value information

We use a crude three point bracketing strategy to locate a minimum of $F(\lambda)$, which is assumed to be unimodal. Three points from the sequence $[w(l-1), w(l), w(l+1)], \quad l = 1, 2, \ldots, l_{max}$ are used, with $w(l) = l\gamma$. Here, the bracketing step size γ is a user selected positive real number. The line search iterations l are incremented until either a minimum is bracketed, or the maximum number of line search iterations l_{max} is exceeded.

We then refine the location of the minimum using a four point (3 interval) golden section search to within a specified user tolerance ξ, or until the maximum number of iterations l_{max} is reached. After each golden section iteration the total interval length is reduced by removal of a sub interval, whereafter a new point is generated within the remaining interval.

The aim therefore is to find $\lambda^{\{k\}}$ for $\xi \in \mathbb{R}$, with $\xi > 0$, such that

$$F(\lambda^{\{k\}} + \xi) = f\left(\boldsymbol{x}^{\{k-1\}} + (\lambda^{\{k\}} + \xi)\boldsymbol{u}^{\{k\}}\right) > F(\lambda^{\{k\}}),$$

and

$$F(\lambda^{\{k\}} - \xi) = f\left(\boldsymbol{x}^{\{k-1\}} + (\lambda^{\{k\}} - \xi)\boldsymbol{u}^{\{k\}}\right) > F(\lambda^{\{k\}}).$$

We will refer to the BFGS algorithm using this function-value-based line search as BFGS(f).

Line searches using only gradient information

This time, we use a two point bracketing strategy to locate a minimum of $F(\lambda)$. Two points from the sequence $[w(l-1), w(l)], \quad l = 1, 2, \ldots, l_{max}$ are used, with $w(l) = l\gamma$. The line search iterations l are incremented until either a sign change in the *associated* derivative $F'^{A}(\lambda)$ is located, or the maximum number of line search iterations l_{max} are reached.

We then refine the location of the sign change in the *associated* derivative using a three point (2 interval) bi-section method to within a specified user tolerance ξ, or until the maximum number of iterations l_{max} is reached. After each bi-section iteration, the total interval width is reduced by half, whereafter a new point is generated in the middle of the remaining interval.

The aim is therefore to find $\lambda^{\{k\}}$ for $\xi \in \mathbb{R}$ with $\xi > 0$, such that

$$F'^{A}(\lambda^{\{k\}} + \xi) = \left[\boldsymbol{\nabla}_A^{\mathrm{T}} f\left(\boldsymbol{x}^{\{k-1\}} + \boldsymbol{u}^{\{k\}}(\lambda^{\{k\}} + \xi)\right)\right] \boldsymbol{u}^{\{k\}} > 0,$$

and

$$F'^{A}(\lambda^{\{k\}} - \xi) = \left[\boldsymbol{\nabla}_A^{\mathrm{T}} f\left(\boldsymbol{x}^{\{k-1\}} + \boldsymbol{u}^{\{k\}}(\lambda^{\{k\}} - \xi)\right)\right] \boldsymbol{u}^{\{k\}} < 0.$$

Note that the gradient-only interval shrinks faster than the classical (function value) interval, as the bi-section interval is reduced by 50%, as opposed to the $\approx 38\%$ of the golden section interval after every iteration. We will refer to the BFGS algorithm with the derivative-based-line-search as BFGS(g).

Termination criteria

Termination criteria need some consideration: if the function values and gradients of an objective or cost function contain step discontinuities, these quantities may not provide robust termination information. Accordingly, we only advocate the robust termination criterion

$$\|\Delta \boldsymbol{x}^{\{k+1\}}\| = \|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| < \epsilon, \tag{3.20}$$

with ϵ small, positive and prescribed. (A maximum number of iterations may of course also be prescribed.)

Algorithmic implementation

Given an initial point $\boldsymbol{x}^{\{0\}}$, the second order line search BFGS method for unconstrained minimization proceeds as follows:

1. **Initialization:** Select real constants $\epsilon > 0$, $\xi > 0$ and $\gamma > 0$. Select integer constants k_{max} and l_{max}. Set $\boldsymbol{G}^{\{0\}} = \boldsymbol{I}$. Set $k := 1$ and $l := 0$.

2. **Gradient evaluation:** Compute $\nabla_A f(\boldsymbol{x}^{\{k\}})$.

3. **Update the search direction $\boldsymbol{u}^{\{k+1\}}$** using (3.16).

4. **Initiate an inner loop to conduct line search:** Find $\lambda^{\{k+1\}}$ using one of the line search strategies described in Section 3.4.1.

5. **Test for reinitialization of $\boldsymbol{G}^{\{k\}}$:** if k mod $n = 0$ then $\boldsymbol{G}^{\{k\}} = \boldsymbol{I}$ else update $\boldsymbol{G}^{\{k\}}$ using (3.17).

6. **Move to the new iterate:** Set $\boldsymbol{x}^{\{k+1\}} := \boldsymbol{x}^{\{k\}} + \lambda^{\{k+1\}}\boldsymbol{u}^{\{k+1\}}$.

7. **Convergence test:** if $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \leq \epsilon$ OR $k = k_{\max}$, stop.

8. **Initiate an additional outer loop:** Set $k := k + 1$ and goto Step 2.

3.4.2 Sequential spherical approximations (SSA)

In sequential approximation optimization (SAO) methods, the approximation functions used can easily be formulated using truncated Taylor expansions in which the curvature may be approximated using function values and gradient information, e.g. Snyman and Hay [52] and Groenwold *et al.* [22]. For illustrative purposes, we will construct two spherical quadratic approximations for use in SAO algorithms. For the first, we use both function value and gradient information for approximating the curvature; for the second, we use gradient information only.

In both cases, we begin with the second order Taylor series expansion of a function f around some current iterate $\boldsymbol{x}^{\{k\}}$, given by

$$\tilde{f}^{\{k\}}(\boldsymbol{x}) = f(\boldsymbol{x}^{\{k\}}) + \nabla_A^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}\boldsymbol{H}^{\{k\}}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}), \qquad (3.21)$$

where superscript k represents an iteration number, $\tilde{f}$ the second order Taylor series approximation to f, ∇_A the associated gradient operator and $\boldsymbol{H}^{\{k\}}$ the Hessian. $f(\boldsymbol{x}^{\{k\}})$ and $\nabla_A f(\boldsymbol{x}^{\{k\}})$ respectively represent the function value and associated gradient vector at the current iterate $\boldsymbol{x}^{\{k\}}$.

For the sake of brevity and simplicity, we will now restrict ourselves to *spherical* approximations to the Hessian $\boldsymbol{H}^{\{k\}}$. Hence, the approximate Hessian or curvature is of the form $\boldsymbol{H}^{\{k\}} = c^{\{k\}}\boldsymbol{I}$, with $c^{\{k\}}$ a scalar, and $\boldsymbol{I}$ the identity matrix. This gives

$$\tilde{f}^{\{k\}}(\boldsymbol{x}) = f(\boldsymbol{x}^{\{k\}}) + \nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}) + \frac{c^{\{k\}}}{2}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}), \qquad (3.22)$$

with the scalar curvature $c^{\{k\}}$ unknown. We will return to the computation of $c^{\{k\}}$ shortly.

Since the sequential approximate subproblems are continuous they may be solved analytically; the minimizer of subproblem k follows from setting the gradient of (3.22) equal to $\mathbf{0}$ to give

$$\boldsymbol{x}^{\{k*\}} = \boldsymbol{x}^{\{k\}} - \frac{\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})}{c^{\{k\}}}. \tag{3.23}$$

In SAO, termination and convergence may be effected through the notion of conservatism. Here, we start with a note on the structure of the spherical approximation functions: they are separable. If these approximation functions are also strictly convex then convergence is guaranteed for a sequence of SAO iterates using conservatism [58]. Arguably, this results in the simplest algorithmic implementation for which termination is guaranteed. Therefore the minimizer of the subproblem $\boldsymbol{x}^{\{k*\}}$ is accepted i.e. $\boldsymbol{x}^{\{k+1\}} = \boldsymbol{x}^{\{k*\}}$ only if $\boldsymbol{x}^{\{k*\}}$ yields a conservative point. We will revisit implementations of conservatism shortly.

Using function value information

If historic function value information is exploited, it is possible to solve for $c^{\{k\}}$ by enforcing $\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) = f(\boldsymbol{x}^{\{k-1\}})$, which results in

$$f(\boldsymbol{x}^{\{k-1\}}) = f(\boldsymbol{x}^{\{k\}}) + \boldsymbol{\nabla}_A^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) + \frac{c^{\{k\}}}{2}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}), \tag{3.24}$$

e.g. see Snyman and Hay [56]. The scalar $c^{\{k\}}$ is then obtained as

$$c^{\{k\}} = \frac{2\left(f(\boldsymbol{x}^{\{k-1\}}) - f(\boldsymbol{x}^{\{k\}})\right)}{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})} - \frac{2\boldsymbol{\nabla}_A^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})}{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})}. \tag{3.25}$$

To ensure that approximation (3.22) is strictly convex, we will herein enforce $c^{\{k\}} = \max(\beta, c^{\{k\}})$, with $\beta > 0$ small and prescribed.

Classical conservatism is solely based on function values, for which Svanberg [58] demonstrated that the SAO sequence $k = 1, 2, \cdots$ will terminate at the minimizer $\boldsymbol{x}^* \leftrightarrow f^*$, if each k^{th} approximation $\tilde{f}(\boldsymbol{x}^{\{k*\}})$ is conservative, i.e. if

$$f(\boldsymbol{x}^{\{k*\}}) \leq \tilde{f}(\boldsymbol{x}^{\{k*\}}). \tag{3.26}$$

We will refer to this SSA algorithm, using conservatism defined by (3.26) and the termination criteria discussed in Section 3.4.1, as SSA(f). We note that the principle behind (3.26) is that updates are only accepted for which the real quality measure ($f(\boldsymbol{x}^{\{k*\}})$ in this instance) is guaranteed to be less than or equal to the approximate quality measure ($\tilde{f}(\boldsymbol{x}^{\{k*\}})$).

Using only gradient information

Approximations to the gradient field may be constructed by simply taking the derivatives of (3.21), which gives

$$\boldsymbol{\nabla}\tilde{f}^{\{k\}}(\boldsymbol{x}) = \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}) + \boldsymbol{H}^{\{k\}}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}). \tag{3.27}$$

At $\boldsymbol{x} = \boldsymbol{x}^{\{k\}}$, the gradients of the function f and the approximation function $\tilde{f}$ match exactly. The approximate Hessian $\boldsymbol{H}^{\{k\}}$ of the approximation $\tilde{f}$ is chosen to match additional information. Here we again only consider the case of a spherical quadratic approximation, where $\boldsymbol{H}^{\{k\}} = c^{\{k\}}\boldsymbol{I}$. $c^{\{k\}}$ is obtained by matching the gradient vector at $\boldsymbol{x}^{\{k-1\}}$. Since only a single free parameter $c^{\{k\}}$ is available, the n components of the respective gradient vectors are matched in a least square sense.

The least squares error is given by

$$E^{\{k\}} = (\boldsymbol{\nabla}\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}}))^{\mathrm{T}}(\boldsymbol{\nabla}\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}})), \tag{3.28}$$

which, after substitution of (3.27) into (3.28), gives

$$\begin{aligned} E^{\{k\}} = (\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}}))^{\mathrm{T}}(\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}) \\ + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}})). \end{aligned} \tag{3.29}$$

Minimization of the least squares error $E^{\{k\}}$ w.r.t. $c^{\{k\}}$ then gives

$$\begin{aligned} \frac{dE^{\{k\}}}{dc^{\{k\}}} = (\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}}))^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) \\ + (\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \boldsymbol{\nabla}_A(\boldsymbol{x}^{\{k-1\}})) = 0, \end{aligned} \tag{3.30}$$

hence

$$c^{\{k\}} = \frac{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k-1\}}) - \boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}}))}{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})}. \tag{3.31}$$

Again, to ensure that approximation (3.22) is strictly convex, we enforce $c^{\{k\}} = \max(\beta, c^{\{k\}})$, with $\beta > 0$ small and prescribed.

We now introduce conservatism, effected using only gradient information. At iterate k, the update is given by $\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}$. This update represents descent of $f(\boldsymbol{x})$ if the projection of the actual function gradient $\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k*\}})$ onto the update direction $(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}})$ is negative, i.e. if

$$\boldsymbol{\nabla}_A^{\mathrm{T}} f(\boldsymbol{x}^{\{k*\}})(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}) \leq \boldsymbol{\nabla}_A^{\mathrm{T}} \tilde{f}(\boldsymbol{x}^{\{k*\}})(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}) = 0. \tag{3.32}$$

Accordingly, any gradient-only approximation may be defined as conservative if (3.32)

holds.

Our gradient-only definition of conservatism is similar in spirit to Svanberg's function value based definition given in (3.26). We only accept updates $\boldsymbol{x}^{\{k*\}}$ for which we can guarantee that the real quality measure $(\boldsymbol{\nabla}_A^{\mathrm{T}} f(\boldsymbol{x}^{\{k*\}})(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}))$ for gradient-only problems) is less than or equal to the approximate quality measure $(\boldsymbol{\nabla}_A^{\mathrm{T}} \tilde{f}(\boldsymbol{x}^{\{k*\}})(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}))$. Although no formal proofs are presented in this study, we can show that our definition of conservatism guarantees convergence for certain classes of functions e.g. smooth convex functions. However, for discontinuous functions in general our definition is not sufficient to guarantee convergence. The challenge however lies in establishing whether a particular practical engineering problem is part of an associated convergent class of discontinuous functions or not. Although we lack strong theoretical evidence for convergence in general, we note that in our experience this gradient-only definition of conservatism suffices to achieve convergence for practical engineering problems.

We will refer to this SSA algorithm, using conservatism defined by (3.32) and the scalar curvature given by (3.31), as SSA(g).

Algorithmic implementation

Given an initial point $\boldsymbol{x}^{\{0\}}$, a {gradient-only}/classical conservative algorithm based on convex separable spherical quadratic approximations for unconstrained minimization proceeds as follows:

1. **Initialization:** Select real constants $\epsilon > 0$, $\alpha > 1$ and initial curvature $c^{\{0\}} > 0$. Set $k := 1$, $l := 0$.

2. **Gradient evaluation:** Compute $\{\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})\}/f(\boldsymbol{x}^{\{k\}})$ and $\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})$.

3. **Approximate optimization:** Construct local approximate subproblem {(3.27)}/(3.22) at $\boldsymbol{x}^{\{k\}}$, using {(3.31)}/(3.25). (In an inner loop, use $c^{\{k\}}$ as calculated in Step 6(b)). Solve this subproblem analytically, to arrive at $\boldsymbol{x}^{\{k*\}}$.

4. **Evaluation:** Compute $\{\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k*\}})\}/f(\boldsymbol{x}^{\{k*\}})$.

5. **Test if $\boldsymbol{x}^{\{k*\}}$ is acceptable:** if {(3.32)}/(3.26) is satisfied, goto Step 7.

6. **Initiate an inner loop to effect conservatism:**

   (a) Set $l := l + 1$.

   (b) Set $c^{\{k\}} := \alpha c^{\{k\}}$.

   (c) Goto Step 3.

7. **Move to the new iterate:** Set $\boldsymbol{x}^{\{k+1\}} := \boldsymbol{x}^{\{k*\}}$.

8. **Convergence test:** if $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \leq \epsilon$, OR $k = k_{max}$, stop.

9. **Initiate an additional outer loop:** Set $k := k + 1$ and goto Step 2.

3.5 Example problems

We now present three example problems to demonstrate the robustness of gradient-only optimization when compared with classical gradient-based optimization in the presence of step discontinuities.

We deliberately choose a small bracketing step size in an attempt to mimic an exact line search. This choice allows us to verify numerically for our examples whether these numerical step discontinuous functions are robustly optimizable in the multidimensional search space. The high number of required inner loops for the BFGS algorithms should therefore be interpreted with this in mind and not be misinterpreted as inherent to line search descent methods or the BFGS algorithm. The performance of the BFGS algorithms herein can readily be enhanced by considering for example an interpolation line search strategy, inexact line searches or just simply appropriate step sizes.

In addition we choose to illustrate an expected difference between gradient-only optimization and classical gradient-based optimization using sequential approximation methods. These methods are well suited to optimize objective functions that have local minima that do not dominate an underlying global function trend, which are reminiscent to the functions we consider in this study.

In all the examples presented here we use remeshing whenever spatial discretizations are required and adaptive time stepping whenever temporal discretizations are required.

3.5.1 Numerical settings

The settings used for the BFGS(f) and BFGS(g) algorithms discussed in Section 3.4.1 are as follows:

- the line search convergence tolerance $\xi = 10^{-5}$,
- the maximum number of line search iterations $l_{max} = 1000$, and
- the bracketing step size $\gamma = 10^{-1}$, except for the material identification study, when we use $\gamma = 10^{-2}$.

For the SSA(f) and SSA(g) algorithms discussed in Section 3.4.2, the settings used are as follows:

- the curvature factor $\alpha = 2$, and
- the initial curvature $c^{\{0\}} = 1$.

Throughout, we use a convergence tolerance $\epsilon = 10^{-4}$, and a maximum number of outer iterations $k_{max} = 500$.

3.5.2 Example problem: temporal and spatial partial differential equations

We consider the sizing design of a fin subjected to a constant volume constraint. The base of the fin initially experiences steady state conditions, whereafter it is exposed to a sinusoidal surge in heat flux. The objective is to minimize some average measure of temperature T over the base of the fin.

For the sizing problem we consider an array of fins of which we only depict half a fin in Figure 3.2, due to symmetry. The two design variables for the sizing problem are the height t_h and width t_w of the triangular part of the fin denoted by $\boldsymbol{x}^{\mathrm{T}} = [x_1 \quad x_2]^{\mathrm{T}} = [t_h \quad t_w]^{\mathrm{T}}$. The temperature field $\boldsymbol{T}$ of the transient problem is solved using the finite element method (FEM) which allows for the construction of the objective function $f(\boldsymbol{x})$. The objective function is the average nodal temperature over the base of the fin after $t_f = 2$ seconds, i.e. $f(\boldsymbol{x}) = \bar{T}_b(\boldsymbol{x})$. The sizing problem is subjected to an equality constraint of constant lateral surface area $b_w b_h + 0.5 t_w (t_h - b_h) = A_0$. Hence, we can solve for x_2 in terms of x_1 from the equality constraint which renders the problem unconstrained in the single variable x_1.

The fin has has constant unit thickness and constant lateral surface area $A_0 = 300\text{mm}^2$. The base of the fin has a fixed width $b_w = 25$ mm, and fixed height $b_h = 4$ mm.

We generate the meshes required for the FEM using a quadratically convergent remeshing strategy [66] that is based on a scheme proposed by Persson and Strang [42]. The resulting meshes have a *varying number of nodes and nodal connectivity*, which induces jump discontinuities into the objective function, since the objective function is formulated in terms of the mesh-dependent[3] temperature field $\boldsymbol{T}$.

Initially, the nodal temperatures $\boldsymbol{T}$ in the fin are the steady state temperatures $\boldsymbol{T}_0$ resulting from a steady state heat flux input $\boldsymbol{q}(0) = \boldsymbol{q}_0 = 30 \times 10^3$ Wm^{-2} at the base of the fin, solved using the approximate FE equations

$$\boldsymbol{K}\boldsymbol{T}_0 = \boldsymbol{q}_0. \tag{3.33}$$

The matrix $\boldsymbol{K}$ is partitioned into contributions from thermal conductivity (we have used $w = 100$ W m^{-1} K^{-1}), and surface convection (we have used $h = 100$ Wm^{-2}K^{-1}), at a constant ambient temperature $T_a = 300$ K.

[3]The mesh generator described in [66] uses linear strain triangle (LST) elements, and we have used an ideal element edge length of $h_0 = 3$.

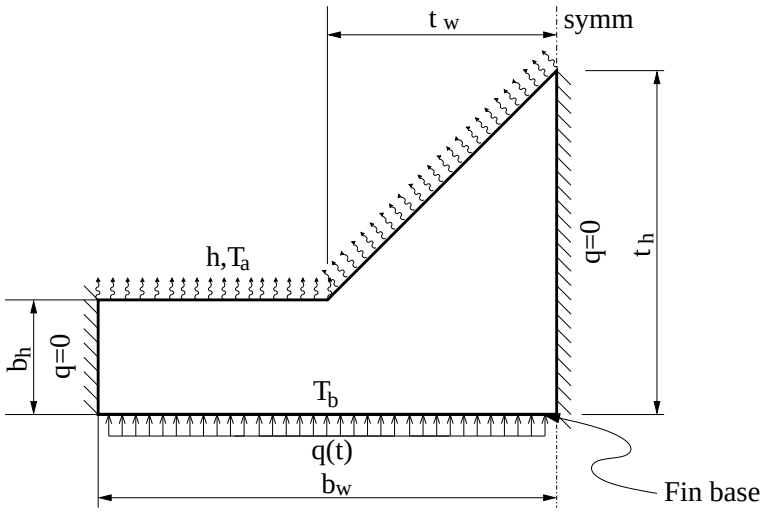


Figure 3.2: Fin model with a uniform time varying heat flux $q(t)$ input at the base, top surface convection with constant convection coefficient h and ambient temperature T_a. The design variables for the sizing problem are the width t_w and height t_h of the triangular part of the fin.

The time dependent heat flux $\boldsymbol{q}(t)$ is given by

$$\boldsymbol{q}(t) = \begin{cases} \boldsymbol{q}_0 + \boldsymbol{q}_p \sin\left(\frac{\pi t}{t_f}\right), & 0 \leq t \leq t_f \\ \boldsymbol{q}_0, & t > t_f \end{cases} \tag{3.34}$$

with $t_f = 2$ s and $q_p = 3 \times 10^5$ Wm^{-2}.

The semi-discrete transient heat equation is given by

$$\boldsymbol{C}\dot{\boldsymbol{T}} + \boldsymbol{K}\boldsymbol{T} = \boldsymbol{q}, \tag{3.35}$$

with $\dot{\boldsymbol{T}}$ denoting the first time derivative of the nodal temperatures $\boldsymbol{T}$ and the $\boldsymbol{C}$ matrix containing contributions from specific heat with a specific heat capacity of 450 J kg^{-1} K^{-1} and material density of 2770 kg m^{-3}. We solve (3.35) using a fully implicit backward Euler finite difference scheme, given by

$$(\boldsymbol{C} + \Delta t^{\{i+1\}}\boldsymbol{K})\boldsymbol{T}^{\{i+1\}} = \boldsymbol{C}\boldsymbol{T}^{\{i\}} + \Delta t^{\{i+1\}}\boldsymbol{q}^{\{i+1\}}, \tag{3.36}$$

$i = 0, 1, 2, \ldots$, while $\Delta t^{\{i+1\}}$ indicates the time step $t^{\{i+1\}} - t^{\{i\}}$.

We start the adaptive time stepping sequence using an initial time step of $\Delta t^{\{1\}} = 0.1$. For iteration $\{i+1\}$, we compute the absolute change in average temperature $\Delta \bar{T}_b^{\{i+1\}}$ over the base of the fin using

$$\Delta \bar{T}_b^{\{i+1\}} = |\bar{T}_b^{\{i+1\}} - \bar{T}_b^{\{i\}}|. \tag{3.37}$$

The updated temperature $\boldsymbol{T}^{\{i+1\}}$ is accepted, and the step size $\Delta t^{\{i+2\}}$ is simultaneously

increased by a factor 1.5, if $\bar{T}_b^{\{i+1\}} < 20$ K. Else, $\Delta t^{\{i+1\}}$ is halved, and iteration $\{i+1\}$ is repeated.

Although the algorithms outlined in Sections 3.4.1 and 3.4.2 are developed to optimize multidimensional functions they can nonetheless be applied to 1D functions, without requiring any modifications. Results are summarized in Table 3.1, obtained for a starting point $x = 30$. Figure 3.3 depicts $f(x)$ and $f'(x)$ for $x \in [30,\ 90]$. Note that the inserts in Figure 3.3 highlight the behavior of the objective function $f(x)$ and it's *associated* derivative $f'^A(x)$ in the vicinity of the results in Table 3.1.

Algorithm	$f(x^{\{N_k\}})$	$f'^A(x^{\{N_k\}})$	$x^{\{N_k\}}$	N_k	N_l
BFGS(f)	4.885E+02	-2.804E+00	3.385E+01	3	79
BFGS(g)	4.459E+02	-6.366E-04	7.950E+01	3	522
SSA(f)	4.458E+02	-2.324E-02	7.720E+01	22	48
SSA(g)	4.459E+02	-5.827E-04	7.950E+01	19	16

Table 3.1: Results obtained with BFGS(f), BFGS(g), SSA(f) and SSA(g) for the univariate transient heat transfer problem.

The optimal solution and positive projection point may coincide, as shown by the inserts in Figures 3.3(a) and (b). Note that the optimal solution occurs over a jump discontinuity. Although SSA(f) is able to overcome many of the numerically induced local minima it is clear from Figure 3.3 that both BFGS(f) and SSA(f) converged to such local minima. To the contrary, both BFGS(g) and SSA(g) are able to robustly overcome the numerically induced local minima and terminate at a positive projection point (indicated by a sign change in $f'^A(x)$ for univariate functions). At first glance the reported results for both BFGS(g) and SSA(g) in Table 3.1 seems inferior to SSA(f). However, at closer inspection it is clear that the function value decreases from 445.9 to 445.7 over the discontinuity, by infinitesimally perturbing the gradient-only solution x_g^* to the right. In addition, the gradient-only solution x_g^* and the optimum x^* describes the same design for all practical purposes. It is clear that when the gradient-only solution x_g^* occurs over a discontinuity for univariate functions then two function values could be reported as $f(x_g^*)$, the one obtained by the left-hand limit and the other by the right-hand limit.

At closer inspection it is clear that SSA(f) converged to a local minimum that is not a positive projection point since the sign of $f'(x)$ remains negative around this point. However, a slight perturbation around this point results in an increase in function value.

3.5.3 Example problems: spatial partial differential equations

Next, we consider firstly the *unconstrained* design of an orthotropic Michell-like structure, and secondly the *equality constrained* design of an orthotropic Michell-like structure.

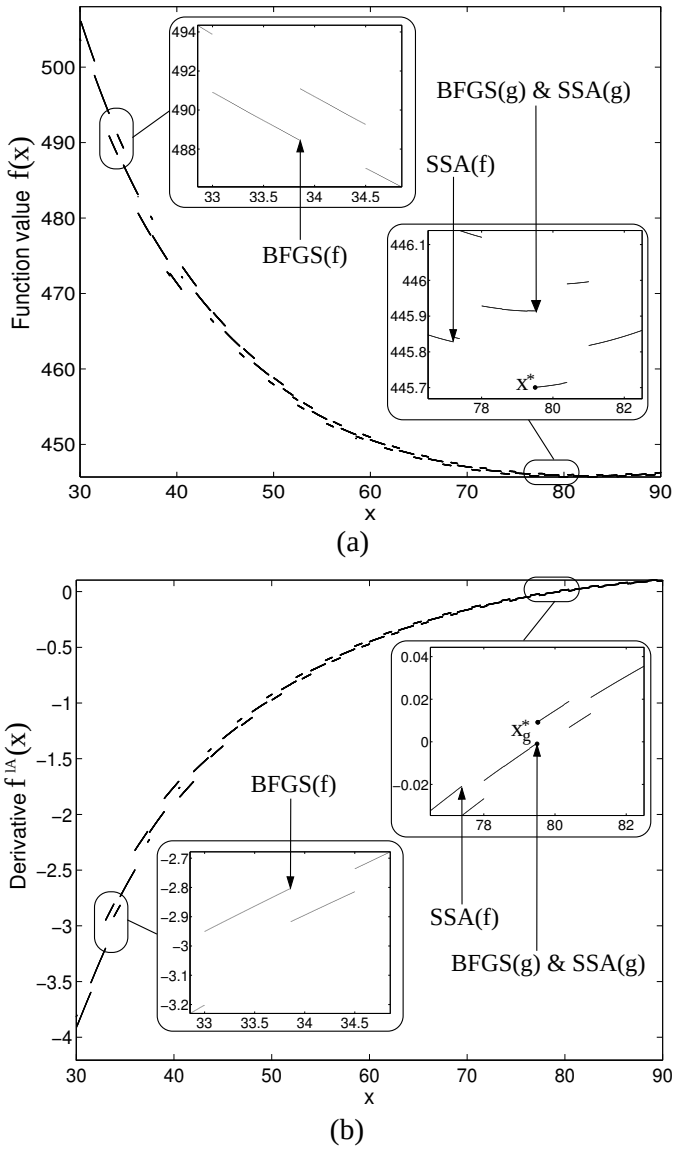


Figure 3.3: (a) Function values, and (b) *associated* derivatives for the univariate transient heat transfer sizing problem. Note the sign change in $f'^{A}(x)$ at $x_g^* = x^* = 79.5$.

Again, both problems are analyzed using methods that are prone to step discontinuities.

The Michell-like structure is depicted in Figure 3.4; the figure depicts the symmetry and support conditions. The structure has a predefined length of 30 mm and thickness of 1 mm. A point load F of 10 N acts at the center bottom of the structure. The boundary of the structure is controlled by the 16 control points $\boldsymbol{x}$ that can only move vertically, which are the 16 design variables in the shape design problem. The control points are linearly spaced along the top and bottom of the Michell-like structure with the first nine control points along the top and the remaining seven along the bottom, as depicted in Figure 3.4. The boundary is linearly interpolated between the control points.

A linear elastic FEM is used to solve the approximate nodal displacement field $\boldsymbol{u}$ from the structural equilibrium PDEs. The spatial discretizations (meshes) for the FEM are

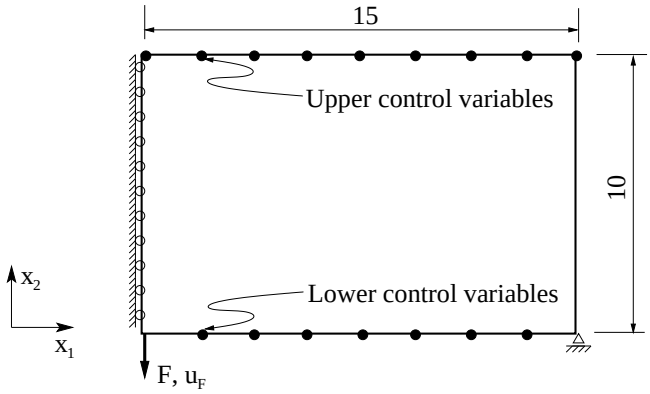


Figure 3.4: Initial geometry of half the Michell-like structure using 16 control points $\boldsymbol{x}$.

again[4] generated using our quadratically convergent remeshing strategy [66].

As in the previous example, the remeshing strategy results in meshes with a varying number of nodes and nodal connectivity, which induces numerical jump discontinuities into the objective function, since the cost function will be formulated in terms of the mesh dependent displacement field $\boldsymbol{u}$.

The nodal displacements $\boldsymbol{u}$ are obtained from the linear elastic approximate finite element equilibrium equations

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{3.38}$$

where $\boldsymbol{K}$ represents the assembled structural stiffness matrix and $\boldsymbol{f}$ the consistent structural loads. Following the usual approach, the system in (3.38) is partitioned along the unknown displacements ($\boldsymbol{u}_{\mathrm{f}}$) and the prescribed displacement ($\boldsymbol{u}_{\mathrm{p}}$), i.e.

$$\boldsymbol{K}\boldsymbol{u} = \begin{bmatrix} \boldsymbol{K}_{\mathrm{ff}} & \boldsymbol{K}_{\mathrm{fp}} \\ \boldsymbol{K}_{\mathrm{pf}} & \boldsymbol{K}_{\mathrm{pp}} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_{\mathrm{f}} \\ \boldsymbol{u}_{\mathrm{p}} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}_{\mathrm{f}} \\ \boldsymbol{f}_{\mathrm{p}} \end{Bmatrix}, \tag{3.39}$$

where $\boldsymbol{f}_{\mathrm{f}}$ represents the prescribed forces and $\boldsymbol{f}_{\mathrm{p}}$ the reactions at the nodes with prescribed displacements. The unknown displacements ($\boldsymbol{u}_{\mathrm{f}}$) are obtained from

$$\boldsymbol{K}_{\mathrm{ff}}\boldsymbol{u}_{\mathrm{f}} = \boldsymbol{f}_{\mathrm{f}} - \boldsymbol{K}_{\mathrm{fp}}\boldsymbol{u}_{\mathrm{p}}. \tag{3.40}$$

The sensitivity of the displacement at the center u_{F} w.r.t $\boldsymbol{x}$ is obtained by direct differentiation of (3.40) as shown in Chapter 2.

The orthotropic stiffness matrix $\boldsymbol{K}$ is computed for Boron-Epoxy in a tape outlay i.e. the fibers are all aligned in a single direction along the longitudinal axis, as indicated by x_1 in Figure 3.4.. We assume plane stress conditions. The orthotropic material properties for Boron-Epoxy used in this study are a Young's modulus along the longitudinal axis of $\mathrm{E}_1 = 228$ GPa, along the transverse axis of $\mathrm{E}_2 = 145$ GPa and a shear modulus of $G_{12} = 48$ GPa. The last independent parameter in classical laminate theory (CLT) is the

[4] This time, we use an ideal element edge length $h_0 = 1$.

Table 3.2: Tabulated results obtained for the unconstrained Michell-like structure.

Algorithm	$f(\boldsymbol{x}^{\{N_k\}})$	$\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{N_k\}})\|$	$\|\Delta\boldsymbol{x}^{\{N_k\}}\|$	N_k	N_l
BFGS(f)	7.740E-01	4.705E-02	0.000E+00	3	60
BFGS(g)	5.941E-01	1.938E-03	0.000E+00	36	717
SSA(f)	6.470E-01	1.602E-02	4.822E-05	15	59
SSA(g)	5.938E-01	1.052E-03	9.582E-05	41	29

Poisson ratio $\nu_{12} = 0.23$, since ν_{21} follows from the symmetry relation $\mathrm{E}_1\nu_{21} = \mathrm{E}_2\nu_{12}$.

Unconstrained shape design of a Michell structure

Consider the unconstrained shape design of the orthotropic Michell structure [20]; we minimize the weighted sum of the displacement and normalized volume of the structure. The cost function is given by

$$f(\boldsymbol{u}, \boldsymbol{x}) = \beta u_\mathrm{F}(\boldsymbol{x}) + f_2(\boldsymbol{x}),$$

where u_F is the displacement at the point of load application of the structure, f_2 is the volume of the structure $V(\boldsymbol{x})$ divided by V_0 and β a weight parameter. Both the displacement at the point of load application $u_\mathrm{F}(\boldsymbol{x})$ and the volume of the structure $V(\boldsymbol{x})$ depend on the design variables $\boldsymbol{x}$. For this study we select $V_0 = 150$ mm^3 and $\beta = 100$.

Results for the unconstrained Michell-like structure The results for the BFGS(f), BFGS(g), SSA(f) and SSA(g) algorithms are summarized in Table 3.2 with the respective final designs depicted in Figures 3.5 (a)-(d). Table 3.2 summarizes the function value $f(\boldsymbol{x}^{\{N_k\}})$, gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{N_k\}})\|$, the norm of the solution update $\|\Delta\boldsymbol{x}^{\{N_k\}}\|$, number of outer iterations N_k as well as the number of inner iterations N_l.

Consider BFGS(f) which converged after 3 outer iterations N_k and, evidently got trapped in a local minimum due to numerical induced step discontinuities. The premature final design is apparent from Figure 3.5 (a).

Similarly, SSA(f) converged after 15 outer iterations N_k also after getting trapped in a step discontinuous minimum. Significant improvements are evident by comparing Figure 3.5 (c) to Figure 3.5 (a), although noticeable improvements could still be made. Clearly, conservative approximation methods are able to overcome *some* step discontinuities and of course even more so when conservatism is relaxed.

Conversely, BFGS(g) and SSA(g) were able to optimize the Michell structure without getting trapped in numerical induced step discontinuities. Consider the similar designs depicted in Figures 3.5 (b) and (d). It is clear that BFGS(g) and SSA(g) improved

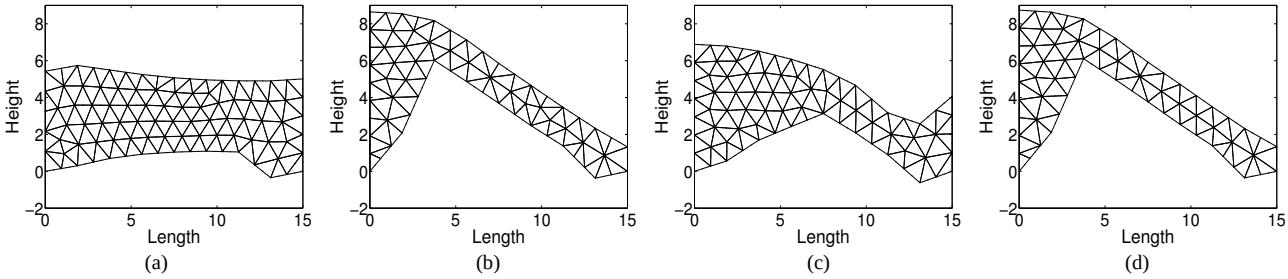


Figure 3.5: Michell-like structure: converged designs obtained with (a) BFGS(f), (b) BFGS(g), (c) SSA(f), and (d) SSA(g).

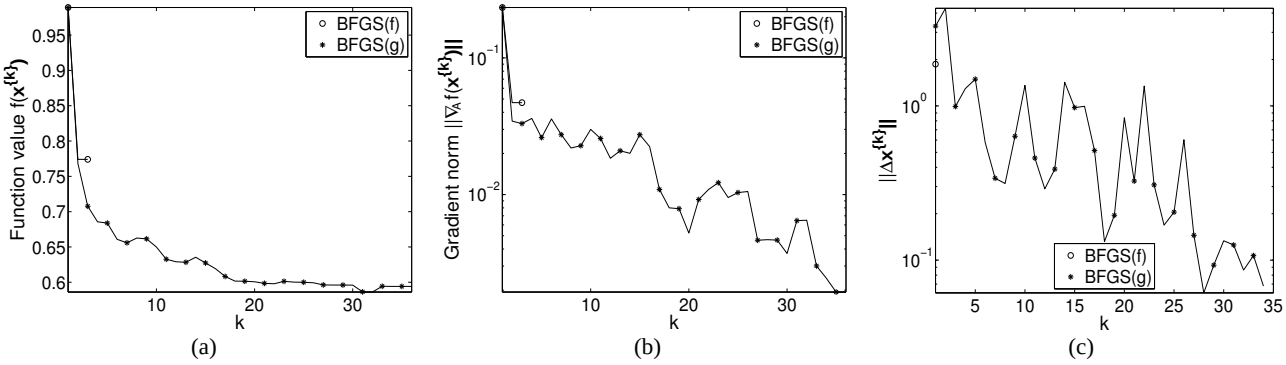


Figure 3.6: Michell-like structure: BFGS(f) and BFGS(g) algorithms convergence history plot of the (a) function value $f(x^{\{k\}})$ and (b) gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})\|$ and (c) the norm of the solution update $\|\Delta \boldsymbol{x}^{\{k\}}\|$.

notably on the designs obtained with BFGS(f) and SSA(f).

We further present for each algorithm their respective convergence histories w.r.t. function value $f(\boldsymbol{x}^{\{k\}})$, gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})\|$ and the norm of the solution update $\|\Delta \boldsymbol{x}^{\{k\}}\|$. The respective histories for the BFGS algorithms are depicted in Figures 3.6 (a)-(c) and for the SSA algorithms in Figures 3.7 (a)-(c).

Monotonic function value decrease for both BFGS(f) and SSA(f) is clearly depicted in respectively Figure 3.6(a) and Figure 3.7(a).

Conversely, non-monotonic function value decrease for both BFGS(g) and SSA(g) is evident in Figure 3.6(a) and Figure 3.7(a).

Constrained shape design of a Michell structure

We now consider the constrained shape design of the orthotropic Michell structure [20]. We minimize the displacement of load application point u_{F} subject to an equality volume constraint $V(\boldsymbol{x}) = V_0$ where V_0 is the prescribed volume of the structure. As indicated, we do so using a Lagrangian formulation.

To solve the dual of the Lagrangian using line search descent methods, we use the BFGS algorithms as described in this chapter except for the following modifications: in the line search strategy every design step $\boldsymbol{x}$ is followed by a multiplier step $(V(\boldsymbol{x}) - V_0)$

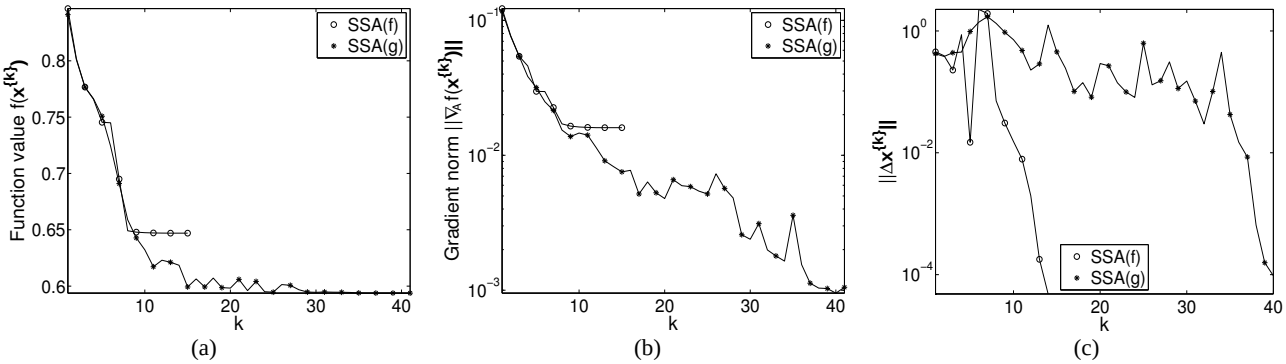


Figure 3.7: Michell-like structure: SSA(f) and SSA(g) algorithms convergence history plot of the (a) function value $f(x^{\{k\}})$ and (b) gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})\|$ and (c) the norm of the solution update $\|\Delta \boldsymbol{x}^{\{k\}}\|$.

Table 3.3: Tabulated results obtained for the constrained Michell-like structure.

Algorithm	$L(\boldsymbol{x}^{\{N_k\}}, \lambda^{\{N_k\}})$	$\|\boldsymbol{\nabla}_A L(\boldsymbol{x}^{\{N_k\}}, \lambda^{\{N_k\}})\|$	$\|\Delta \boldsymbol{x}^{\{N_k\}}\|$	$\|\Delta \lambda^{\{N_k\}}\|$	N_{k}	N_{l}
BFGS(f)	8.239E-01	2.191E-01	0.000E+00	2.459E-01	11	348
BFGS(g)	3.635E-01	5.092E-03	0.000E+00	3.104E-04	53	1188
SSA(f)	3.712E-01	2.992E-02	1.778E-07	1.762E-02	41	129
SSA(g)	3.564E-01	4.292E-03	0.000E+00	2.645E-04	144	238

in the outer loop. In turn, the multiplier λ is kept constant during the inner loop where the bracketing search and golden section refinement occurs.

On the other hand we solve the dual of the Lagrangian using the SSA algorithms as described in this chapter with the following adjustments: every design update $\boldsymbol{x}$ in the outer loop is followed by a multiplier step $(V(\boldsymbol{x}) - V_0)$. In turn, the multiplier λ is kept constant during the inner loops.

Both the BFGS and SSA algorithms are terminated when $\|[\Delta \boldsymbol{x}^{N_k} \ \Delta \lambda^{N_k}]\| < \epsilon$ or, when $\|\Delta \boldsymbol{x}^{N_k}\| < \epsilon$ for five consecutive iterations.

Results for the constrained Michell-like structure The results for the BFGS(f), BFGS(g), SSA(f) and SSA(g) algorithms are summarized in Table 3.3, with the respective final designs depicted in Figures 3.8 (a)-(d). Table 3.3 summarizes the Lagrangian $L(\boldsymbol{x}^{\{N_k\}}, \lambda^{\{N_k\}})$, the norm of the Lagrangian gradient $\|\boldsymbol{\nabla}_A L(\boldsymbol{x}^{\{N_k\}}, \lambda^{\{N_k\}})\|$, the design variable update $\|\Delta \boldsymbol{x}^{\{N_k\}}\|$, the Lagrange multiplier update $\|\Delta \lambda^{\{N_k\}}\|$, the number of outer iterations N_k, and the number of inner iterations N_l.

BFGS(f) converged to a numerically induced local minimum after 11 outer iterations N_k, with the final design depicted in Figure 3.8 (a). Similarly, SSA(f) converged to a numerically induced local minimum after 41 outer iterations N_k, with the final design depicted in Figure 3.8 (c). As shown in Table 3.3, the respective Lagrange multiplier updates $\|\Delta \lambda^{\{N_k\}}\|$ are orders larger than the design variable updates $\|\Delta \boldsymbol{x}^{\{N_k\}}\|$, since both

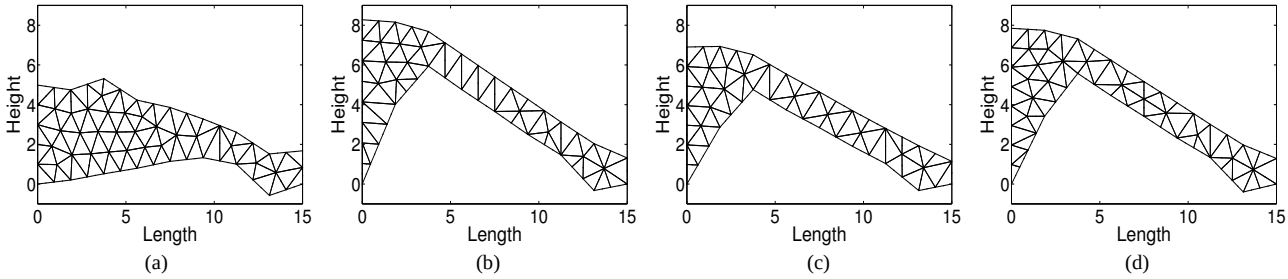


Figure 3.8: Michell-like structure: converged designs obtained with (a) BFGS(f), (b) BFGS(g), (c) SSA(f), and (d) SSA(g).

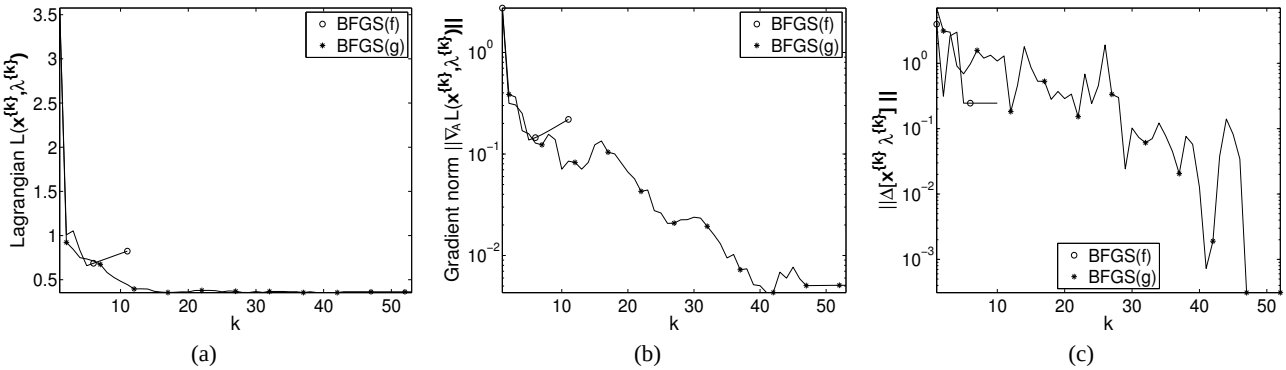


Figure 3.9: Michell-like structure: BFGS(f) and BFGS(g) algorithms convergence history plot of (a) the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, (b) the norm of the Lagrangian gradient $\|\boldsymbol{\nabla}_A L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})\|$ and (c) the norm of the solution update $\|\Delta[\boldsymbol{x}^{\{k\}}\ \lambda^{\{k\}}]\|$.

algorithms got stuck in numerically induced local minima for five consecutive iterations, while significantly violating the equality constraint.

Conversely, BFGS(g) and SSA(g) were able to optimize the Michell structure with the respective designs depicted in Figures 3.8 (b) and (d). It is clear that BFGS(g) and SSA(g) improved notably on the designs obtained with BFGS(f) and SSA(f). As shown in Table 3.3, the Lagrange multiplier updates $\|\Delta\lambda^{\{N_k\}}\|$ are small.

We further present for each algorithm their respective histories w.r.t. the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, the norm of the Lagrangian gradient $\|\boldsymbol{\nabla}_A L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})\|$ and the norm of the solution update $\|\Delta[\boldsymbol{x}^{\{k\}}\ \lambda^{\{k\}}]\|$. The respective histories for the BFGS algorithms are depicted in Figures 3.9 (a)-(c) and for the SSA algorithms in Figures 3.10 (a)-(c).

3.5.4 Example problems: temporal partial differential equations

To illustrate the advantages of gradient-only optimization for cost functions formulated from temporal PDEs discretized using non-constant strategies, we finally perform a material identification study. The aim of the material identification study is to find the parameter values of a modified Voce model (law) [33] which best describe experimentally measured yield stress data. We achieve this by merely minimizing the least squares er-

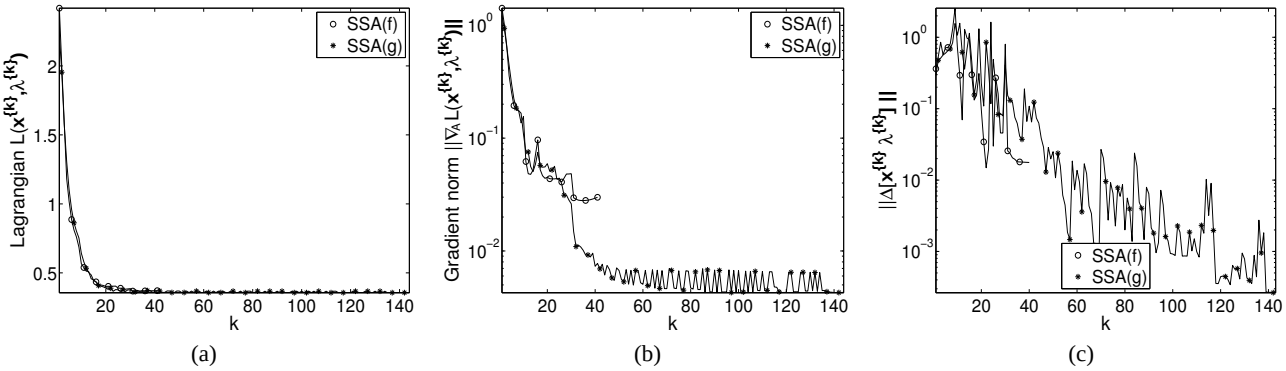


Figure 3.10: Michell-like structure: convergence histories for SSA(f) and SSA(g) of (a) the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, (b) the norm of the Lagrangian gradient $\|\boldsymbol{\nabla}_A L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})\|$ and (c) the norm of the solution update $\|\Delta[\boldsymbol{x}^{\{k\}}\ \lambda^{\{k\}}]\|$.

ror between the experimental data and the modified Voce model data points. Although inverse problems are usually ill-posed we obtained promising results without regularizing the problem.

The Voce model [31, 62] is given by

$$\frac{d\sigma_y}{d\epsilon_p} = \theta_0 \left(1 - \frac{\sigma_y}{\sigma_{ys}}\right), \tag{3.41}$$

where σ_y represents the evolving yield stress, ϵ_p the plastic strain, σ_{ys} the saturation stress and θ_0 the extrapolated strain hardening rate for zero flow stress.

Although the Voce model describes yield stress evolution to moderate strains adequately, it usually has poor validity at large strains. To overcome large strain deficiencies, we use a modified Voce model to capture linear hardening behavior observed at larger strains. The Voce model is modified to include a stage IV evolution equation [33],

$$\frac{d\sigma_4}{d\epsilon_p} = c, \tag{3.42}$$

to obtain

$$\frac{d\sigma_y}{d\epsilon_p} = \theta_0 \left(1 - \frac{\sigma_y}{\sigma_{ys}} + \frac{\sigma_4}{\sigma_y}\right). \tag{3.43}$$

We calculate σ_y for a specific value of ϵ_p by numerically integrating (3.42) and (3.43), using a forward Euler method in which we first solve for the stress contribution from the stage IV evolution equation given by

$$\sigma_4^{\{i+1\}} = \sigma_4^{\{i\}} + c\Delta\epsilon_p^{\{i\}}. \tag{3.44}$$

Here, the superscript $\{i\}$ indicates the iteration number and $\Delta\epsilon_p^{\{i\}}$ the plastic strain step size at iteration $\{i\}$, i.e. $\epsilon_p^{\{i+1\}} - \epsilon_p^{\{i\}}$.

The updated evolving yield stress is then computed using

$$\sigma_y^{\{i+1\}} = \sigma_y^{\{i\}} + \theta_0 \left(1 - \frac{\sigma_y^{\{i\}}}{\sigma_{ys}} + \frac{\sigma_4^{\{i+1\}}}{\sigma_y^{\{i\}}}\right) \Delta\epsilon_p^{\{i\}}. \tag{3.45}$$

Each first order system of DEs requires an initial condition at $\{i\} = 0$. To allow for optimization flexibility, the initial conditions for $\sigma_4^{\{0\}}$ and $\sigma_y^{\{0\}}$ are chosen to be design variables. $\Delta\epsilon_p^{\{i\}}$ is adjusted using an adaptive time step scheme.

We start with an initial $\Delta\epsilon_p^0 = 10^{-3}$ in the adaptive time step scheme. For each iteration $\{i\}$ we compute $\Delta\sigma_y^{\{i\}}$, i.e. $\sigma_y^{\{i+1\}} - \sigma_y^{\{i\}}$. The step size $\Delta\epsilon_p^{\{i+1\}}$ increases by a factor 1.5 from iteration $\{i+1\}$ to $\{i+2\}$, unless $\Delta\sigma_y^{\{i\}}$ is more than a defined maximum allowable evolved stress update $\Delta\sigma_y^{\max} = 10$ MPa. We then half $\Delta\epsilon_p^{\{i\}}$ and redo the $\{i+1\}^{\text{th}}$ iteration. During the entire numerical integration procedure, we limit the step size $\Delta\epsilon_p^{\{i\}}$ between a maximum allowable step $\Delta\epsilon_p^{\max} = 10^{-1}$ and a minimum allowable step $\Delta\epsilon_p^{\min} = 10^{-3}$.

Let us consider the experimental measured data taken at various plastic strain points. The experimental data may range from a few, to many hundreds of points at arbitrarily spaced intervals, depending on the experimental setup. The resulting discrete model points in turn have their own arbitrarily spaced intervals. Some interpolation strategy is required since the experimental data points and the modified Voce model data points may not always coincide. Three obvious linear interpolation strategies that come to mind are interpolating

- the experimental data points to the model data points,
- the model data points to the experimental data points, and
- both the model and experimental data points to intermediate points.

In this study we only consider the first interpolation strategy. The experimental data required for the inverse problem is obtained by solving the modified Voce model with the parameter values given in Table 3.4. This allows for a zero error solution. We identify the following five design variables: $\boldsymbol{x} = [x_1, x_2, \ldots, x_5] = [c, \theta_0, \sigma_{ys}, \sigma_4^{\{0\}}, \sigma_y^{\{0\}}]$, to minimize the least squares error between the experimental and model data points.

In order to improve the variable scaling we normalize the design vectors by the optimum $\boldsymbol{x}^*$ given in Table 3.4, although the results are only presented as non-normalized variables.

Finally, we consider the interpolation of the experimental data points to the model data points. We therefore linearly interpolate between the experimentally measured data to obtain experimental data for the corresponding plastic strains in the numerical model. The disadvantage being that the number of experimental data points used depends on the time step sequence of the numerical model.

Table 3.4: Parameter values used to construct experimental data for the inverse problem using the modified Voce model with the adaptive time step algorithm.

c	θ_0	σ_{ys}	$\sigma_4^{\{0\}}$	$\sigma_y^{\{0\}}$
90	1000	180	0.5	1

Table 3.5: Tabulated results for the least squares fit between the modified Voce model data points and the linearly interpolated experimental data points.

Algorithm	$f(\boldsymbol{x}^{\{N_k\}})$	$\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{N_k\}})\|$	$\|\Delta\boldsymbol{x}^{\{N_k\}}\|$	N_k	N_l
BFGS(f)	2.605E+03	9.284E+03	0.000E+00	3	35
BFGS(g)	1.981E-04	3.200E-01	0.000E+00	36	1695
SSA(f)	1.392E+01	1.220E+03	7.172E-05	16	39
SSA(g)	3.939E-01	7.206E+00	9.086E-05	22	8

The cost function of the design problem is given by

$$f(\boldsymbol{x}) = \sum_{j=1}^{r} (d_{ei}^j - d_m^j)^2, \tag{3.46}$$

where r is the number of model data points i.e. the numerical integration sequence, d_{ei}^j is the linearly interpolated yield stress from the experimental data and d_m^j is the yield stress obtained from the numerical model. Details regarding the computation of the analytical sensitivities for this problem are given in Section 6 of the Appendix.

Results obtained for interpolated experimental data The results obtained with the BFGS(f), BFGS(g), SSA(f) and SSA(g) algorithms are summarized in Table 3.5. The table presents the function value $f(\boldsymbol{x}^{\{N_k\}})$, the gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{N_k\}})\|$, the norm of the solution update $\|\Delta\boldsymbol{x}^{\{N_k\}}\|$, the number of outer iterations N_k and the number of inner iterations N_l.

From Table 3.5 it is clear that both BFGS(f) and SSA(f) converged to suboptimal local minima, in particular when considering the large norm of the gradient at the solutions. In turn, BFGS(g) and SSA(g) obtained solutions with the norms of the gradient considerably lower at the converged solution. To investigate the nature of the local minimum of BFGS(f) we depict the function value and *associated* derivative (around the converged solution) along the final search direction in Figures 3.11 (a) and (b) respectively. Conversely, BFGS(g) and SSA(g) were able to effectively minimize the least squares error. We also depict for BFGS(g) the function value and *associated* derivative (around the gradient projection point) along the final search direction in Figures 3.12 (a) and (b) respectively.

Table 3.6: Final designs obtained for the least squares fit between the modified Voce model data points and the linearly interpolated experimental data points.

	c	θ_0	σ_{ys}	$\sigma_4^{\{0\}}$	$\sigma_y^{\{0\}}$
$\boldsymbol{x}^*$	90	1000	180	0.5	1
BFGS(f)	69.88	1453.97	148.52	0.60	1.20
BFGS(g)	90.85	999.49	179.74	0.50	1.00
SSA(f)	71.12	1025.31	181.32	0.59	1.20
SSA(g)	71.39	1004.91	186.17	0.59	1.20

Consider the history plots for BFGS(f) and BFGS(g) depicted in Figures 3.13 (a)-(c): BFGS(f) result in a *monotonic* decrease in function value $f(\boldsymbol{x}^{\{k\}})$, as opposed to the *non-monotonic* decrease of BFGS(g). The associated gradient norm is depicted in Figure 3.13 (b). The distance from the optimum $\|\boldsymbol{x}^* - \boldsymbol{x}^{\{k\}}\|$, depicted in Figure 3.13 (c), shows how BFGS(g) approaches the optimum.

The history plots for SSA(f) and SSA(g) are depicted in Figures 3.14 (a)-(c) which illustrates that a *monotonic* decrease in function value is obtained by SSA(f) as opposed to a *non-monotonic* decrease for SSA(g). The distance from the optimum $\|\boldsymbol{x}^* - \boldsymbol{x}^{\{k\}}\|$ depicted in Figure 3.14 (c), indicates how SSA(g) approaches the optimum.

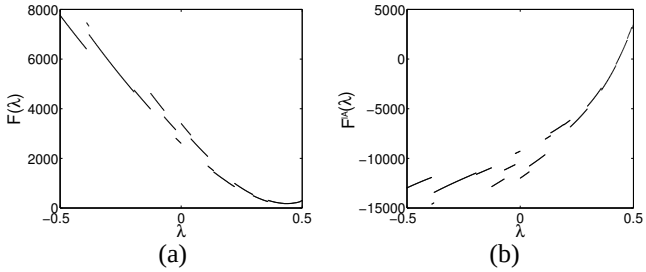


Figure 3.11: Function value and *associated* derivative along the search direction around the optimal point obtained with BFGS(f) for the linearly interpolated experimental data.

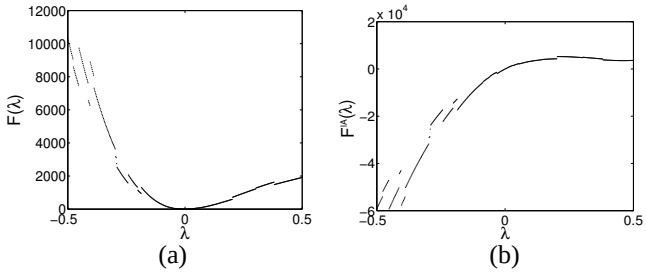


Figure 3.12: Function value and *associated* derivative along the search direction around the solution obtained with BFGS(g) for the linearly interpolated experimental data.

Lastly, we include the final design vectors $\boldsymbol{x}^{\{N_k\}}$ obtained in Table 3.6.

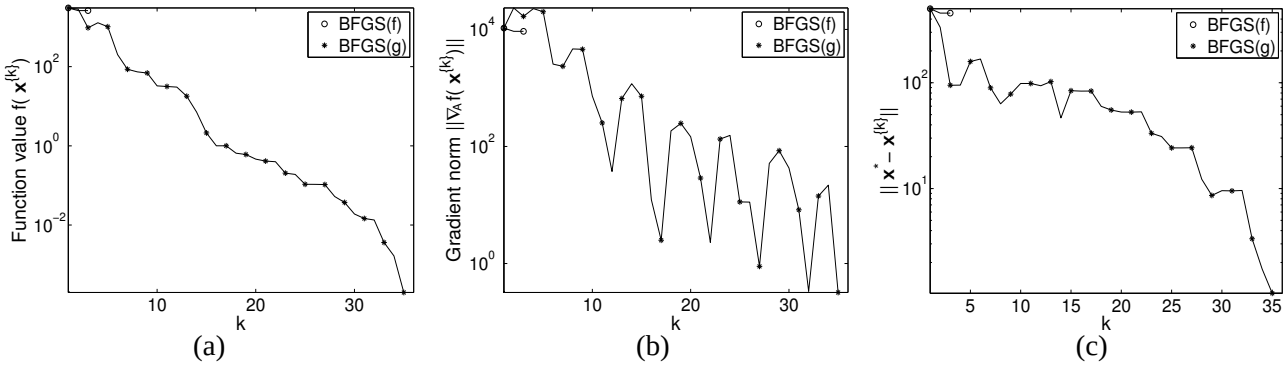


Figure 3.13: Modified Voce model: BFGS(f) and BFGS(g) algorithms convergence history plot of (a) the function value $f(\boldsymbol{x}^{\{k\}})$, (b) the gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})\|$ and (c) the distance from the optimum $\|\boldsymbol{x}^* - \boldsymbol{x}^{\{k\}}\|$, for the linearly interpolated experimental data.

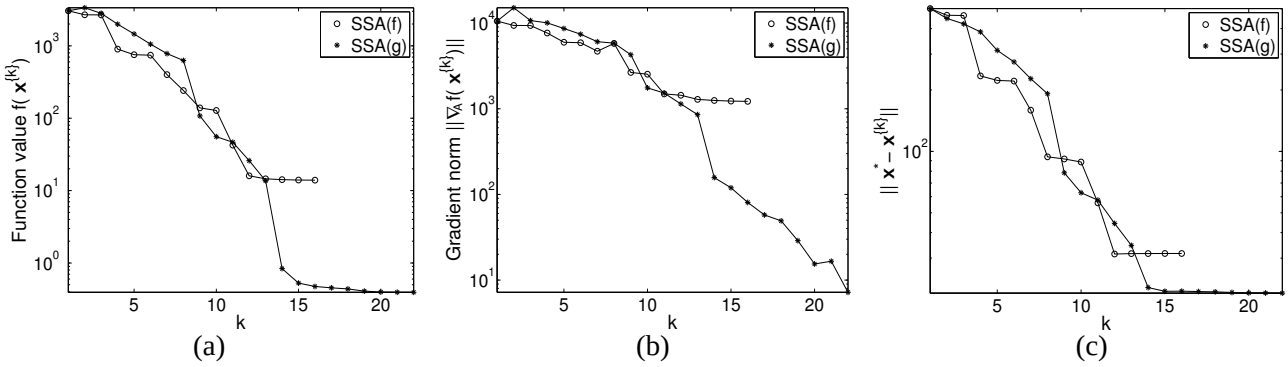


Figure 3.14: Modified Voce model: SSA(f) and SSA(g) algorithms convergence history plot of (a) the function value $f(\boldsymbol{x}^{\{k\}})$, (b) the gradient norm $\|\boldsymbol{\nabla}_A f(\boldsymbol{x}^{\{k\}})\|$ and (c) the distance from the optimum $\|\boldsymbol{x}^* - \boldsymbol{x}^{\{k\}}\|$, for the linearly interpolated experimental data.

3.6 Conclusions

We have studied the minimization of objective functions containing non-physical step or jump discontinuities. These discontinuities arise when (partial) differential equations are discretized using non-constant methods: the functions become discontinuous and non-differentiable at these discontinuities. We can however compute (semi) analytical sensitivities [40] at these discontinuous points since every point has an associated discretization for which such a computation can be performed.

To illustrate, we proposed gradient-only implementations of the BFGS algorithm and a SAO algorithm for discontinuous problems, and applied these algorithms to a selection of problems of practical interest, both unconstrained and constrained. These are the design of a heat exchanger fin, the shape design of an orthotropic Michell-like structure, and a material identification study using a modified Voce law, all discretized using non-constant methods. In each instance, the gradient based algorithms found superior solutions to the classical methods that use both function and gradient information.

As opposed to surrogate methods based on design of experiments techniques, which scale poorly, gradient-only algorithms based on classical optimization algorithms that scale well, may also be expected to scale well (provided the gradient computations scale well); this may well become an important application of gradient-only methods. Another envisaged application of gradient-only algorithms are any problem for which gradient computations are inexpensive.

CHAPTER 4

Theory of gradient-only optimization

In this chapter we consider some theoretical aspects of gradient-only optimization for the unconstrained optimization of objective functions containing non-physical step or jump discontinuities. The (discontinuous) gradients are however assumed to be accurate and everywhere uniquely defined. This kind of discontinuity indeed arises when the optimization problem is based on the solutions of systems of partial differential equations, when variable discretization techniques are used (remeshing in spatial domains or variable time stepping in temporal domains). These discontinuities, which may cause local minima, are artifacts of the numerical strategies used and should not influence the solution to the optimization problem. We demonstrate that it is indeed possible to ignore these local minima due to discontinuities, if only gradient information is used. Various gradient-only algorithmic options are discussed. The implications are that variable discretization strategies, so important in the numerical solution of partial differential equations, can be combined with efficient local optimization algorithms.

This chapter is organized as follows: We give an introduction to discontinuous objective functions in Section 4.1. We then define an optimization problem and solution to the problem that is solely based on the gradient of a function in Section 4.2. In Section 4.3, we introduce the gradient-only optimization problem and in Section 4.4 we offer proofs of convergence of descent sequences defined in the previous section. We give practical considerations regarding gradient-only optimization algorithms in Section 4.5, and present a brief comparative discussion of classical mathematical programming and gradient-only optimization in Section 4.6. In Section 4.7 we present a shape optimization problem of practical importance, and a number of analytical test functions. Concluding remarks then follow.

4.1 Introduction

In this study we consider theoretical aspects regarding gradient-only approaches to avoid spurious (local) minima for unconstrained optimization. Here, gradient-only optimization algorithms refers to optimization strategies that solely considers first order information of a scalar (cost) objective function in computing update directions and update step lengths.

Many problems in engineering and the applied sciences are described by (partial) differential equations (P)DEs, e.g. Newton's second law, Poisson's equations, Maxwell's electromagnetic equations, the Black-Scholes equations, the Lotka-Volterra equations and Einstein's field equations. Analytical solutions to these are seldom available and in many cases, (approximate) numerical solutions need to be computed. These (approximate) numerical solutions are often obtained by employing discretization methods e.g. finite difference, finite element and finite volume methods.

(P)DEs also often describe the physics of some optimization problem. These optimization problems are usually numerically approximated that results in an approximate optimization problem, which are then optimized using numerical optimization techniques. During optimization the domain over which the (P)DEs are solved may remain constant but the discretization may be required to change to ensure convergence or efficiency of the solution e.g. integrating over a fixed time domain using variable time steps. Alternatively, the design variables may describe the domain over which the (P)DEs are solved. A change in design variables therefore change the solution domain, which in turn requires the discretization to change e.g. shape optimization.

In order to affect these discretization changes, we distinguish between two classes of strategies. First, constant discretization strategies continuously adjusts a reference discretization when the solution domain change (and hence generates a fixed discretization if the solution domain remains fixed). Secondly, variable discretization strategies generate new independent discretizations irrespective of whether or not the solution domain changes. For example, temporal (P)DEs may be solved using fixed or variable time steps. For spatial PDEs, the equivalents are fixed and mesh movement strategies on the one hand, and remeshing on the other. Fixed time steps and mesh movement strategies however may imply serious difficulties, e.g. impaired convergence rates and highly distorted grids and meshes, which may even result in failure of the computational procedures used. The variable discretization strategies are preferable by far.

One consequence of using variable discretization while solving an optimization problem, is that the resulting objective functions contain discontinuities.

Accordingly, the optimization of piece-wise smooth step discontinuous functions usually requires highly specialized optimization strategies, and possibly, heuristic approaches. In contrast, constant discretization strategies result in smooth, continuous objective functions that present no significant challenges to optimization algorithms.

It therefore appears that the analyst has two options, namely i) combining an efficient local optimization algorithms with non-ideal constant discretization, or ii) combining a less efficient global optimization algorithm with ideal non-constant discretization.

Hence, variable time step methods and remeshing techniques are normally avoided in optimization, due to the very fact that the required global optimization algorithms are prohibitively expensive. An important spatial example is structural shape optimization, in which fixed or mesh movement strategies are almost always used; the very motivation for this being that remeshing strategies cannot be used efficiently, due to the induced non-physical local minima during optimization, e.g. see References [1, 9, 32, 39].

To avoid confusion, we emphasize the difference between physical discontinuities that occur in the solution domain of PDEs, such as shear banding in plasticity and shock waves in supersonic flow, and non-physical discontinuities. The non-physical discontinuities we refer to occur in the objective function of an optimization problem as opposed to discontinuities in the solution of a PDE.

As we aim to develop a theoretical framework for gradient-only optimization in this chapter we will invariably restrict ourselves to various classes of objective functions in our discussions and analysis through the course of this chapter. In addition, we restrict ourselves to unconstrained optimization and note that some practical constrained optimization problems can be successfully reformulated as unconstrained optimization problems using a penalty formulation. Consider the following unconstrained minimization problem: find the minimizer $\boldsymbol{x}^*$ of a real-valued function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$, such that

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}),\ \forall\ \boldsymbol{x} \in X, \tag{4.1}$$

with X the convex set of all possible solutions. If the function f is strictly convex, coercive and at least twice continuously differentiable, i.e. $f \in C^2$, the minimizer $\boldsymbol{x}^* \in X$ is characterized by the optimality criterion $\boldsymbol{\nabla} f(\boldsymbol{x}^*) = \boldsymbol{0}$, with the Hessian matrix $\boldsymbol{H}(\boldsymbol{x}^*)$ positive semi-definite at $\boldsymbol{x}^*$. Here, $\boldsymbol{\nabla}$ represents the gradient operator. For this programming problem, many well known minimization algorithms are available, e.g. steepest descent, (preconditioned) conjugate gradient methods and quasi-Newton methods like BFGS. However, if f is discontinuous, i.e. $f \notin C^0$, the minimizer $\boldsymbol{x}^*$ of the mathematical programming problem (4.1) may not satisfy the optimality criterion given above. Indeed, the optimality criterion may not even be defined in the classical sense although it may be defined using generalized gradients (subdifferential) $\partial f(\boldsymbol{x})$ [12, 13, 17] which requires $\boldsymbol{0} \in \partial f(\boldsymbol{x}^*)$.

To the best of our knowledge, only a few gradient-only optimization algorithms have been developed, see References [4, 43, 44, 49, 53, 54, 64]. This includes the widely known gradient-only Newton's method [64], which locates diminishing gradients.. A notable contribution on the optimization of functions that are *not* everywhere uniquely

differentiable (non-differentiable) is the subgradient methods [49]. Subgradient methods reduce to steepest descent algorithms with step lengths *a priori* determined i.e. the line searches do not depend on any computed information during optimization, when an objective function is continuously differentiable. All of these algorithms are used to minimize objective functions and require the condition that the gradient $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$ or the subdifferential[1] $\partial f(\boldsymbol{x}^*)$ must contain $\mathbf{0}$ depending on the differentiability of $f(\boldsymbol{x})$ at $\boldsymbol{x}^*$. Accordingly, the well-known efficient optimization algorithms mentioned above may be unable to find $\boldsymbol{x}_g^*$ since they are concerned with obtaining a minimizer $\boldsymbol{x}^*$ for an objective function [17].

In turn, if it is known or assumed that $\boldsymbol{x}_g^*$ coincides with a minimizer $\boldsymbol{x}^*$ of $f(\boldsymbol{x})$ then these problems can be approached from a classical mathematical programming perspective which have to be solved using global optimization algorithms. This is due to the numerically induced step discontinuities that manifest as local minima in the function domain. We however show that if it is known that accurate *associated gradient information* that is *everywhere* defined is available, the resulting discontinuous problems may still be optimized efficiently since gradient-only optimization ignores these numerically induced step discontinuities. Gradient-only optimization therefore transforms a problem plagued with numerically induced local minima to a problem free from it. Recall that a third option now becomes available to the analyst: combine an efficient local optimization algorithm with ideal non-constant discretization.

Let us first present two illustrative examples of non-physical step discontinuities, to set the tone for this chapter. The first is rather trivial, the second not quite.

4.1.1 Univariate example problem: Newton's cooling law

Consider Newton's law of cooling, which states that the rate of heat loss of a body is proportional to the difference in temperature between a body and the surroundings of that body, given by the linear first order DE:

$$\frac{dT}{dt} = -\kappa(T(t) - T_{\text{env}}), \tag{4.2}$$

with the well known analytical solution

$$T(t) = T_{\text{env}} + (T_{\text{init}} - T_{\text{env}})e^{-\kappa t}. \tag{4.3}$$

Here κ is a positive proportionality constant, $T(t)$ the temperature of the body at time t and T_{env} the temperature of the surroundings of the body.

We consider the temperature $T(t)$ of a body after 1s, for $0.5 \leq \kappa \leq 2$, with $T(0) = 100°\text{C}$ at $t = 0$, and $T_{\text{env}} = 10°\text{C}$ for all t. The analytical solution of the

[1] A subdifferential is the set of subgradients at a point.

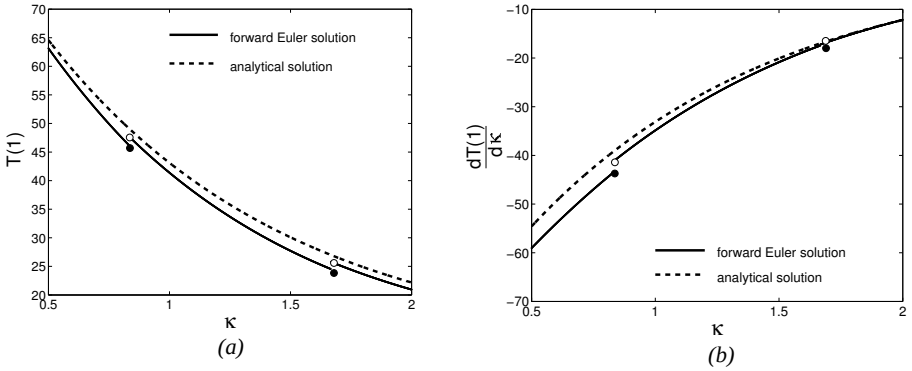


Figure 4.1: Numerical and analytical solutions for Newton's cooling law. (a) Temperature T after 1 second for $0.5 \leq \kappa \leq 2$, and (b) the corresponding *associated derivative* $\frac{dT(1)}{d\kappa}$.

bodies temperature $T(1)$ is depicted in Figure 4.1(a) and the first *associated derivative* of $T(1)$ w.r.t. κ is depicted in Figure 4.1(b).

Solving Eq. (4.2) for $0.5 \leq \kappa \leq 2$ with a forward Euler method using a variable time stepping strategy introduces step discontinuities in the temperature response; this is shown in Figure 4.1(a). For the variable time step strategy we decrease the time step whenever an allowed temperature increment is exceeded, otherwise we gradually increase the time step. The corresponding discontinuous derivatives are plotted in Figure 4.1(b). Note that although discontinuous, the *associated derivatives* are everywhere uniquely defined for the numerically computed objective function.

4.1.2 Multivariate example problem: Shape optimization

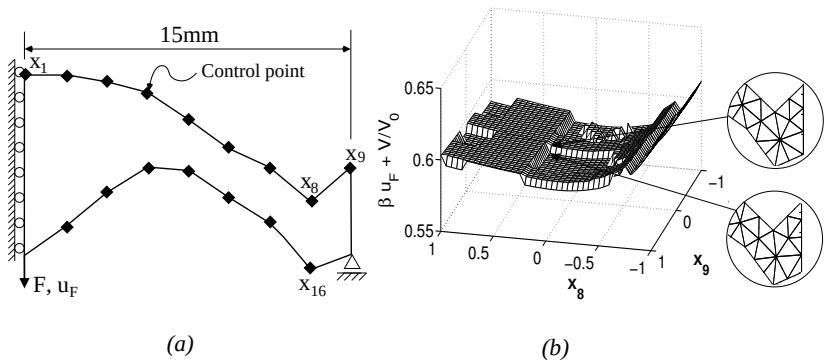


Figure 4.2: (a) Structure, boundary conditions and control variables and (b) the vertical displacement u_{F} for variations of the two rightmost upper control variables (x_8, x_9) for the Michell shape optimization problem.

Next, we consider a non-trivial benchmark problem in structural shape optimization, namely the so-called Michell structure [20] depicted in Figure 4.2(a). The geometry is represented using 16 control variables that are linearly spaced horizontally with only

vertical degrees of freedom and piecewise linear interpolation between control points. The objective of this shape optimization problem is to minimize the sum of the weighted vertical displacement βu_{F} at the point of load application and normalized volume $\frac{V}{V_0}$ for a unit thickness structure with F = 1N, $V_0 = 150\text{mm}^3$ and $\beta = 1$. The displacement u_{F} is computed using a linear elastic finite element method with linear strain triangular elements (e.g. see [15]). For the material properties we use Young's modulus $E = 200\text{GPa}$ and Poisson's ratio $\nu = 0.3$. The meshes required for the finite element analyses are generated using a quadratic convergent remeshing strategy [66] with ideal element length $h_0 = 1\text{mm}$. To illustrate the discontinuous nature of the objective function, the two control variables x_8 and x_9 are perturbed around the reference configuration depicted in Figure 4.2(a) over the range -1.0 through 1.0, using constant intervals of 0.05.

The resulting objective function values are shown in Figure 4.2(b). The step discontinuities due to remeshing are clearly evident; they result since the number of nodes, and the nodal connectivity, changes. This is evident from Figure 4.2(b): a small decrease in x_9 results in 3 elements (top insert in Figure 4.2(b)) as opposed to 4 elements (bottom insert Figure 4.2(b)) on the rightmost edge of the structure.

4.1.3 Introductory comments

Clearly, the introduced non-physical discontinuities cannot be accommodated in optimization methods developed for C^1 continuous objective functions. However, again note that the *associated gradients* of the piece-wise smooth step discontinuous functions considered in this study are everywhere uniquely defined. Consider the positive projection point x_g^* that occurs over a discontinuity as depicted by $f_N(x)$ in Figure 3.1, with a piece-wise smooth part L of the function to the left and a piece-wise smooth part R of the function to the right of it. Both the left and the right hand limits represent approximations to the analytical value of the objective function; the left and right hand limits differ only as a result of the *discretization technique* used, and these values approach each other in the limit of mesh refinement anyway. Hence, the value of the objective function being reported is not unique.

In this study, we consider the unconstrained optimization of objective functions containing non-physical step or jump discontinuities with accurate *associated gradients* that are everywhere defined. For the sake of brevity, we restrict our efforts to unconstrained optimization (but the implications for constrained optimization are clear).

4.2 Definitions

Not all step discontinuities are necessarily problematic for classical optimization, and we distinguish between two step discontinuity types, namely those that are *inconsistent*

with the function *trend*, and those that are *consistent* with the function *trend*, as shown in Figure 4.3. (All other discontinuities may be taken to be representable of either a minimum or a maximum.) To represent semi-continuity of f we introduce a double empty/filled circle convention as depicted in Figure 4.3(a), where a filled circle indicates $F(\lambda_0)$. Upper semi-continuity is represented by the filled/empty circle pair indicated by 1's in Figure 4.3(a) i.e. the filled/empty circles lie *above* f. Lower semi-continuity in turn is represented by the empty/filled circle pair indicated by 2's, i.e. the empty/filled circles lie *below* f, in Figure 4.3(a).

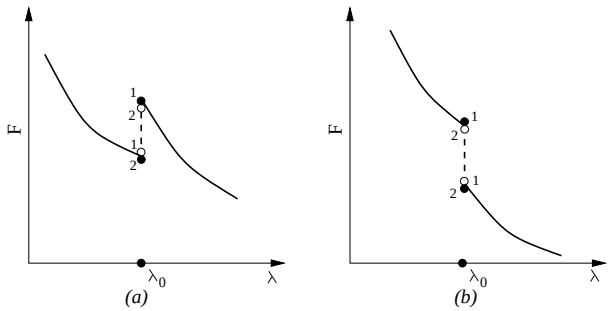


Figure 4.3: Upper and lower semi-continuous univariate functions with (a) an inconsistent step discontinuity, and (b) a consistent step discontinuity.

Figure 4.3(a) depicts an inconsistent step discontinuity; the function decreases as λ increases, but the step discontinuity results in an increase of the function over the step discontinuity. Similarly, Figure 4.3(b) depicts a consistent step discontinuity.

The functions we consider in this study are step discontinuous and therefore not everywhere differentiable. However computationally the derivatives and gradients are everywhere computable since the analysis is per se restricted to the part of the objective function before, or after a discontinuity. We therefore define an *associated derivative* $f'^{A}(x)$ and *associated gradient* $\nabla_A f(\boldsymbol{x})$ which follow computationally when the sensitivity analysis is consistent [48]. Firstly, we define the *associated derivative*

Definition 4.2.1. Let $f : X \subset \mathbb{R} \to \mathbb{R}$ be a real *univariate* piece-wise smooth step discontinuous function that is everywhere defined. The *associated derivative* $f'^{A}(x)$ for $f(x)$ at a point x is given by the derivative of $f(x)$ at x when $f(x)$ is differentiable at x. The *associated derivative* f'^{A} for $f(x)$ non-differentiable at x, is given by the left-sided derivative of $f(x)$ when x is associated to the left piece-wise continuous section of the discontinuity, otherwise it is given by the right-sided derivative.

Secondly, the *associated gradient* is defined as follows:

Definition 4.2.2. Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real *multivariate* piece-wise smooth step discontinuous function that is everywhere defined. The *associated gradient* $\nabla_A f(\boldsymbol{x})$ for $f(\boldsymbol{x})$ at a point $\boldsymbol{x}$ is given by the gradient of $f(\boldsymbol{x})$ at $\boldsymbol{x}$ when $f(\boldsymbol{x})$ is differentiable at $\boldsymbol{x}$. The *associated gradient* $\nabla_A f(\boldsymbol{x})$ if $f(\boldsymbol{x})$ is non-differentiable at $\boldsymbol{x}$ is defined as the

vector of partial derivatives where each partial derivative is an *associated derivative* (see Definition 4.2.1).

It follows from Definitions 4.2.1 and 4.2.2 that the *associated gradient* reduces to the gradient of a function that is everywhere differentiable.

We now proceed to develop a self-contained theoretical framework for gradient-only optimization. Although what follows is rather straightforward extensions of classical concepts, it is included for the sake of completeness.

Definition 4.2.3. Let $f : (a, b) \subset \mathbb{R} \to \mathbb{R}$ be a real univariate function that is not necessarily continuous in both function value $f(\lambda)$ and *associated derivative* $f'^{A}(\lambda)$ but for which $f(\lambda)$ and $f'^{A}(\lambda)$ are uniquely defined for every $\lambda \in (a, b)$. Then, $f(\lambda)$ is said to have a (resp., strictly) negative *associated derivative* on (a, b) if $f'^{A}(\lambda)$ (resp., $<$) ≤ 0, $\forall\ \lambda \in (a, b)$, e.g. see Figure 4.3. Conversely, $f(\lambda)$ is said to have a (resp., strictly) *positive associated derivative* on (a, b) if $f'^{A}(\lambda)$ (resp., $>$) ≥ 0, $\forall\ \lambda \in (a, b)$.

Next, we define lower and upper semi-continuity of the *associated gradient*.

Definition 4.2.4. Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real valued function with an *associated gradient* field $\nabla_A f(\boldsymbol{x})$ that is uniquely defined for every $\boldsymbol{x} \in X$.

- Then the *associated directional derivative* along a normalized direction $\boldsymbol{u} \in \mathbb{R}^n$ is lower semi-continuous at $\boldsymbol{y} \in X$ if

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{y})\boldsymbol{u} \leq \liminf_{\lambda \to 0^{\pm}} \nabla_A{}^{\mathrm{T}} f(\boldsymbol{y} + \lambda \boldsymbol{u})\boldsymbol{u},\ \lambda \in \mathbb{R}.$$

- The *associated directional derivative* along a normalized direction $\boldsymbol{u} \in \mathbb{R}^n$ is upper semi-continuous at $\boldsymbol{y} \in X$ if

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{y})\boldsymbol{u} \geq \limsup_{\lambda \to 0^{\pm}} \nabla_A{}^{\mathrm{T}} f(\boldsymbol{y} + \lambda \boldsymbol{u})\boldsymbol{u},\ \lambda \in \mathbb{R}.$$

- The *associated directional derivative* along a normalized direction $\boldsymbol{u} \in \mathbb{R}^n$ is pseudo-continuous at $\boldsymbol{y} \in \mathbb{R}^n$ if it is both upper and lower semi-continuous at $\boldsymbol{y}$.

We note that a univariate function $f(\lambda)$ may be step discontinuous at a point $\bar{\lambda} \in (a, b)$, but the *associated derivative* may still be pseudo-continuous at $\bar{\lambda}$, e.g. the function

$$f(\lambda) = \begin{cases} \lambda^2, & \lambda < -1 \\ \lambda^2 - 2, & \lambda \geq -1 \end{cases},$$

is not pseudo-continuous at $\bar{\lambda} = 1$. However, the *associated derivative*

$$f'^{A}(\lambda) = \begin{cases} 2\lambda, & \lambda < -1 \\ 2\lambda, & \lambda \geq -1 \end{cases},$$

is pseudo-continuous at $\bar{\lambda} = -1$, where we defined the *associated derivative* at $\bar{\lambda} = -1$ by the right-hand limit.

4.3 Gradient-only optimization problem

We now present the general unconstrained gradient-only optimization problem that is equivalent to the classical minimization problem presented in Section 4.1 for smooth convex cost functions.

Problem 4.3.1. *Given a real-valued function $f : X \subset \mathbb{R}^n \to \mathbb{R}$, find a non-negative associated gradient projection point $\boldsymbol{x}_g^* \in X$ such that for every $\boldsymbol{u} \in \{\boldsymbol{y} \in \mathbb{R}^n \,/\, \|\boldsymbol{y}\| = 1\}$ there exists a real number $r_u > 0$, and the following holds:*

$$\nabla_A{}^T f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} \geq 0 \; \forall \; \lambda \in (0, r_u].$$

Accordingly, we define (resp. non-negative) non-positive generalized associated gradient projection points that characterizes a solution to imply a (resp. minimum) maximum according to the *associated gradient* field of a scalar function, be it local or global, as follows:

Definition 4.3.1. Suppose that $f : X \subset \mathbb{R}^n \to \mathbb{R}$ is a real-valued function for which the *associated gradient* field $\nabla_A f(\boldsymbol{x})$ is uniquely defined for every $\boldsymbol{x} \in X$.

Then, a point $\boldsymbol{x}_g^* \in X$ is a generalized non-negative associated gradient projection point (G-NN-GPP) if there exists a real number $r_u > 0$ for every $\boldsymbol{u} \in \{\boldsymbol{y} \in \mathbb{R}^n \,/\, \|\boldsymbol{y}\| = 1\}$ such that

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} \geq 0, \; \forall \; \lambda \in (0, r_u].$$

Similarly, a point $\boldsymbol{x}_g^* \in X$ is a generalized non-positive associated gradient projection (G-NP-GPP) point if there exists a real number $r_u > 0$ for every $\boldsymbol{u} \in \{\boldsymbol{y} \in \mathbb{R}^n \,/\, \|\boldsymbol{y}\| = 1\}$ such that

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} \leq 0, \; \forall \; \lambda \in (0, r_u].$$

A special case of Problem 4.3.1 is given below which we refer to as the strict unconstrained gradient-only optimization problem.

Problem 4.3.2. *Given a real-valued function $f : X \subset \mathbb{R}^n \to \mathbb{R}$, find a $\boldsymbol{x}_g^* \in X$ such that for every $\boldsymbol{u} \in \{\boldsymbol{y} \in \mathbb{R}^n \,/\, \|\boldsymbol{y}\| = 1\}$ there exists a real number $r_u > 0$, and the following holds:*

$$\nabla_A{}^T f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} > 0 \; \forall \; \lambda \in (0, r_u].$$

Accordingly, we define strict *associated gradient projection points* (resp. non-negative/non-positive) to imply a (resp. minimum/maximum) according to the *associated gradient* field of a scalar function, be it local or global, as follows:

Definition 4.3.2. Suppose that $f : X \subset \mathbb{R}^n \to \mathbb{R}$ is a real-valued function for which the *associated gradient* field $\nabla_A f(\boldsymbol{x})$ is uniquely defined for every $\boldsymbol{x} \in X$.

Then, a point $\boldsymbol{x}_g^* \in X$ is a strict non-negative associated gradient projection point (S-NN-GPP) if there exists a real number $r_u > 0$ for every $\boldsymbol{u} \in \{\boldsymbol{y} \in \mathbb{R}^n \ / \ \|\boldsymbol{y}\| = 1\}$ such that

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} > 0, \ \forall \ \lambda \in (0, r_u].$$

Similarly, a point $\boldsymbol{x}_g^* \in X$ is a strict non-positive associated gradient projection point (S-NP-GPP) if there exists a real number $r_u > 0$ for every $\boldsymbol{u} \in \ \{\boldsymbol{y} \in \mathbb{R}^n \ / \ \|\boldsymbol{y}\| = 1\}$ such that

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} < 0, \ \forall \ \lambda \in (0, r_u].$$

It follows that the strict unconstrained gradient-only optimization problem is included in the generalized unconstrained gradient-only optimization problem.

We now show that our definition for a generalized non-negative associated gradient projection point (G-NN-GPP) is consistent with the classical mathematical programming (MP) definition of a minimizer. To do so we consider the *associated gradient* at a G-NN-GPP for C^1 continuous functions.

Proposition 4.3.3. *Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be continuous with continuous first partial derivatives around a generalized non-negative associated gradient projection point (G-NN-GPP) $\boldsymbol{x}_g^* \in X$. Then, $\nabla_A f(\boldsymbol{x}_g^*) = \boldsymbol{0}$.*

Proof. By Definition 4.3.1 of a G-NN-GPP, $\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} \geq 0 \ \forall \ \boldsymbol{u}$ and $\lambda > 0$ sufficiently small. Also for all corresponding $-\boldsymbol{u}$, $\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda(-\boldsymbol{u}))(-\boldsymbol{u}) \geq 0$. Consequently since $f(\boldsymbol{x}) \in C^1$,

$$\lim_{\lambda \to 0} \nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* + \lambda \boldsymbol{u})\boldsymbol{u} = \nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^*)\boldsymbol{u} \geq 0$$

and

$$\lim_{\lambda \to 0} \nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^* - \lambda \boldsymbol{u})\boldsymbol{u} = \nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}_g^*)\boldsymbol{u} \leq 0.$$

Thus, since $\boldsymbol{u} \neq \boldsymbol{0}$ is arbitrarily chosen and $\nabla_A f(\boldsymbol{x})$ continuous the above two statements can only be true simultaneously if $\nabla_A f(\boldsymbol{x}_g^*) = \boldsymbol{0}$, which completes the proof. □

4.3.1 Discontinuous gradient projection points (GPP)

Our newly introduced definitions for a non-negative associated gradient projection point (NN-GPP) or a non-positive associated gradient projection point (NP-GPP) of a function only require that the *associated gradient* field be uniquely defined everywhere; no assumptions regarding the continuity of the function are required. Hereafter associated gradient projection point (GPP) or associated gradient projection set (GPS) is used to imply either a non-negative or non-positive associated gradient projection (point / set). We therefore omit the conventional inclusion of a saddle (point / set). In addition the

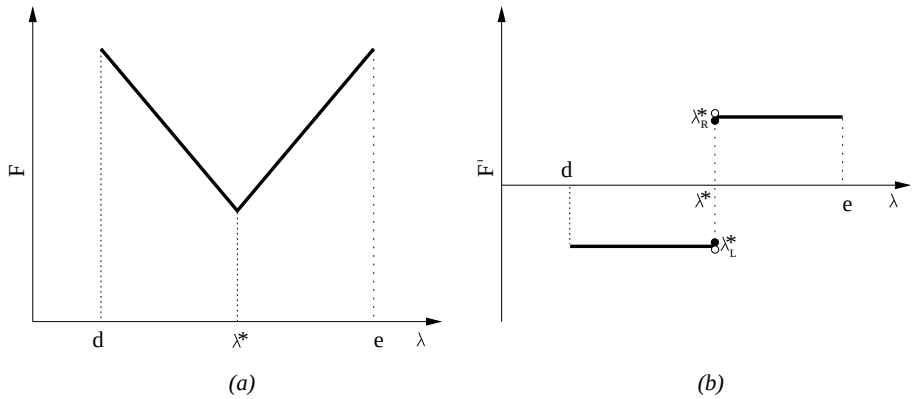


Figure 4.4: An illustration of (a) the function value and (b) the corresponding *associated derivative* that is either upper or lower semi-continuous, with a step discontinuous strict non-negative associated gradient projection point (S-NN-GPP) in $\in (d, e)$.

function may be discontinuous at a GPP. We first consider discontinuous GPPs for univariate functions and then for multivariate functions. An example of a function with a discontinuous NN-GPP is the absolute value function with the *associated derivative* at the minimum point λ^* defined by either the left or right limit as depicted in Figure 4.4, as opposed to the conventional undefined derivative at λ^*. The *associated derivative* at λ^* is therefore either upper or lower semi-continuous as indicated by the double empty/filled notation.

Proposition 4.3.4. *Let $f : [d, e] \subset \mathbb{R} \to \mathbb{R}$ be a real univariate function that is not necessarily continuous in both function value $f(\lambda)$ and* associated derivative *$f'^A(\lambda)$ but for which $f(\lambda)$ and $f'^A(\lambda)$ are uniquely defined for every $\lambda \in [d, e]$. In addition, let $f'^A(\lambda)$ be step discontinuous (upper or lower* associated derivative *semi-continuous) at a (resp. generalized / strict) associated gradient projection point ((resp. G/S)-GPP) $\lambda^* \in (d, e)$ according to Definition (resp. 4.3.1 / 4.3.2). Let λ^*_L be the left limit and λ^*_R the right limit of λ^*.*

Then in addition to the (resp. G/S)-GPP λ^*, either λ^*_L is a (resp. G/S)-GPP if

$$\lim_{\lambda \to \lambda^{*-}} f'^A(\lambda) \neq f'^A(\lambda^*),$$

or λ^*_R is a (resp. G/S)-GPP if

$$\lim_{\lambda \to \lambda^{*+}} f'^A(\lambda) \neq f'^A(\lambda^*).$$

Proof. This is immediate from Definition (resp. 4.3.1 / 4.3.2). □

For multivariate functions we can state a similar proposition.

Proposition 4.3.5. *Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real valued function with* associated gradient *field $\nabla_A f(\boldsymbol{x})$ that is uniquely defined for every $\boldsymbol{x} \in X$. In addition let $\nabla_A f(\boldsymbol{x})$*

be step discontinuous at a (resp. G/S)-GPP $\boldsymbol{x}_g^* \in X$ *according to Definition (resp. 4.3.1 / 4.3.2). Then the limit of every sequence to* $\boldsymbol{x}_g^*$ *is also a (resp. G/S)-GPP if*

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_g^*} \nabla_A f(\boldsymbol{x}) \neq \nabla_A f(\boldsymbol{x}_g^*).$$

Proof. This is immediate from Proposition 4.3.4. □

We now introduce a (resp. generalized / strict) associated gradient projection set ((resp. G/S)-GPS) to accommodate all the (resp. generalized / strict) associated gradient projection points ((resp. G/S)-GPPs) at a (resp. G/S)-GPP $\boldsymbol{x}_g^*$.

Definition 4.3.3. Let $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a real valued function with *associated gradient* field $\nabla_A f(\boldsymbol{x})$ that is uniquely defined for every $\boldsymbol{x} \in X$. In addition let $\boldsymbol{x}_g^* \in X$ be a (resp. G/S)-GPP according to Definition (resp. 4.3.1 / 4.3.2).

We define the set S as follows:

$$S = \left\{ \boldsymbol{x}_g^*, \boldsymbol{y} : \lim_{\boldsymbol{y}\to\boldsymbol{x}_g^*} \nabla_A f(\boldsymbol{y}) \neq \nabla_A f(\boldsymbol{x}_g^*),\ \forall\ \boldsymbol{y} \in \mathbb{R}^n \right\}$$

The set S is then a (resp. generalized / strict) non-negative associated gradient projection set (resp. S_{G-NN} / S_{S-NN}) of $\boldsymbol{x}_g^*$ if every $\boldsymbol{x} \in$ (resp. S_{G-NN} / S_{S-NN}) is a (resp. G/S)-NN-GPP according to Definition (resp. 4.3.1 / 4.3.2).

The set S is then a (resp. generalized / strict) non-positive associated gradient projection set (resp. S_{G-NP} / S_{S-NP}) of $\boldsymbol{x}_g^*$ if every $\boldsymbol{x} \in$ (resp. S_{G-NP} / S_{S-NP}) is a (resp. G/S)-NP-GPP according to Definition (resp. 4.3.1 / 4.3.2).

We now show that our definition of a (resp. generalized / strict) associated gradient projection set (resp. S_G / S_S) is consistent with the classical mathematical programming definition of a minimum or maximum point.

Proposition 4.3.6. *Let* $f : X \subset \mathbb{R}^n \to \mathbb{R}$ *be a real valued function with a continuous* associated gradient *field* $\nabla_A f(\boldsymbol{x})$ *with* $\boldsymbol{x} \in X$. *Then, any (resp. generalized / strict) associated gradient projection set (resp.* S_G / S_S*) of* $\boldsymbol{x}_g^* \in X$ *is a singleton* $\{\boldsymbol{x}_g^*\}$ *such that* $\nabla_A f(\boldsymbol{x}_g^*) = \boldsymbol{0}$.

Proof. It follows from the definition of a (resp. S_G / S_S) given in Definition 4.3.3 that

$$\lim_{\boldsymbol{y}\to\boldsymbol{x}_g^*} \nabla_A f(\boldsymbol{y}) = \nabla_A f(\boldsymbol{x}_g^*),\ \boldsymbol{y} \in \mathbb{R}^n$$

since $f(\boldsymbol{x})$ is C^1 continuous at $\boldsymbol{x}_g^*$ by premise of which (resp. S_G / S_S) is reduced to a singleton.

The second assertion that $\nabla_A f(\boldsymbol{x}_g^*) = \boldsymbol{0}$ follows from Proposition 4.3.3. □

4.3.2 Derivative descent sequences

Now that we have defined GPPs and GPSs solely based on the *associated gradient* field of a function, we proceed to define descent sequences that only considers the *associated gradient* field of a function.

Definition 4.3.4. For a given sequence $\{\boldsymbol{x}^{\{k\}} \in X \subset \mathbb{R}^n : k \in \mathbb{P}\}$ suppose $\nabla_A f(\boldsymbol{x}^{\{k\}}) \neq \mathbf{0}$ for some k and $\boldsymbol{x}^{\{k\}} \notin S_{G-NN}$ with S_{G-NN} defined in Definition 4.3.3. Then the sequence $\{\boldsymbol{x}^{\{k\}}\}$ is an associated derivative descent sequence for $f : X \to \mathbb{R}$, if an associated sequence $\{\boldsymbol{u}^{\{k\}} \in \mathbb{R}^n : k \in \mathbb{P}\}$ may be generated such that if $\boldsymbol{u}^{\{k\}}$ is a descent direction from the set of all possible descent directions at $\boldsymbol{x}^{\{k\}}$, i.e. $\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})\boldsymbol{u}^{\{k\}} < 0$ then

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{\{k+1\}})\boldsymbol{u}^{\{k\}} < 0, \text{ for } \boldsymbol{x}^{\{k\}} \neq \boldsymbol{x}^{\{k+1\}} \tag{4.4}$$

We also include the definition of a stricter class of associated derivative descent sequences which we require for convergence proofs of multimodal functions of dimension two and higher in order to exclude oscillating sequences. Oscillating sequences may occur when the sequence defined in Definition 4.3.4 is considered.

Definition 4.3.5. For a given sequence $\{\boldsymbol{x}^{\{k\}} \in X \subset \mathbb{R}^n : k \in \mathbb{P}\}$ suppose $\nabla_A f(\boldsymbol{x}^{\{k\}}) \neq \mathbf{0}$ for some k and $\boldsymbol{x}^{\{k\}} \notin S_{G-NN}$ with S_{G-NN} defined in Definition 4.3.3. Then the sequence $\{\boldsymbol{x}^{\{k\}}\}$ is a conservative associated derivative descent sequence for $f : X \to \mathbb{R}$, if an associated sequence $\{\boldsymbol{u}^{\{k\}} \in \mathbb{R}^n : k \in \mathbb{P}\}$ may be generated such that if $\boldsymbol{u}^{\{k\}}$ is a descent direction from the set of all possible descent directions at $\boldsymbol{x}^{\{k\}}$ then

$$\nabla_A{}^{\mathrm{T}} f\left(\boldsymbol{x}^{\{k\}} + \lambda(\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}})\right)\boldsymbol{u}^{\{k\}} < 0, \forall\, \lambda \in [0,1] \text{ for } \boldsymbol{x}^{\{k\}} \neq \boldsymbol{x}^{\{k+1\}}. \tag{4.5}$$

4.4 Proofs of convergence for derivative descent sequences

Before we present proofs of convergence of (conservative) associated derivative descent sequences we include two gradient-only definitions of the well-known concepts in classical mathematical programming to simplify our proofs of convergence. First, we present a definition of coercive functions based solely on the *associated gradient* of a function [41]. Although this definition does not bear a strict analogy with the conventional coercive definition it suffices for our purposes.

Definition 4.4.1. Let $\boldsymbol{x}^1, \boldsymbol{x}^2 \in \mathbb{R}^n$. Then a real valued function $f : X \subset \mathbb{R}^n \to \mathbb{R}$ with *associated gradient* field $\nabla_A f(\boldsymbol{x})$ that is uniquely defined for every $\boldsymbol{x} \in X$, is associated

derivative coercive if there exist a positive number R_M such that ${\nabla_A}^{\mathrm{T}} f(\boldsymbol{x}^2)(\boldsymbol{x}^2 - \boldsymbol{x}^1) > \epsilon$ with $\epsilon > 0 \in \mathbb{R}$ for non perpendicular $\nabla_A f(\boldsymbol{x}^2)$ and $(\boldsymbol{x}^2 - \boldsymbol{x}^1)$, whenever $\|\boldsymbol{x}^2\| \geq R_M$ and $\|\boldsymbol{x}^1\| < R_M$.

Secondly, we present definitions for univariate and multivariate associated gradient unimodality based solely on the *associated gradient* field of a real valued function [4].

Definition 4.4.2. A univariate function $f : X \subset \mathbb{R} \to \mathbb{R}$ with associated derivative $f'^A(\lambda)$ uniquely defined for every $\lambda \in X$, is (resp., strictly) associated derivative unimodal over X if there exists a $x_g^* \in X$ such that

$$f'^A(x_g^* + \lambda u)u \geq (\text{resp., } >)\ 0,\ \forall\ \lambda \in \{\beta : \beta > 0 \text{ and } \beta \subset \mathbb{R}\} \\ \text{and}\ \ \forall\ u \in \{-1, 1\} \text{ such that } [x_g^* + \lambda u] \in X. \quad (4.6)$$

We now consider (resp., strictly) associated derivative unimodality for multivariate functions [46].

Definition 4.4.3. A multivariate function $f : X \subset \mathbb{R}^n \to \mathbb{R}$ is (resp., strictly) associated derivative unimodal over X if for all $\boldsymbol{x}^1$ and $\boldsymbol{x}^2 \in X$ and $\boldsymbol{x}^1 \neq \boldsymbol{x}^2$, every corresponding univariate function

$$F(\lambda) = f(\boldsymbol{x}^1 + \lambda(\boldsymbol{x}^2 - \boldsymbol{x}^1)), \quad \lambda \in [0, 1] \subset \mathbb{R}$$

is (resp., strictly) associated derivative unimodal according to Definition 4.4.2.

4.4.1 Univariate functions

Now that we have an *associated derivative* based definition of unimodality for univariate functions we present a proof of convergence for strict univariate associated derivative unimodal functions when associated derivative descent sequences are considered.

Theorem 4.4.1. *Let $f : \Lambda \subseteq \mathbb{R} \to]-\infty, \infty]$ be a univariate function that is strictly associated derivative unimodal as defined in Definition 4.4.2, with first* associated derivative *$f'^A : \Lambda \to]-\infty, \infty[$ uniquely defined everywhere on Λ. If $\lambda^{\{0\}} \in \Lambda$ and $\{\lambda^{\{k\}}\}$ is an associated derivative descent sequence, as defined in Definition 4.3.4, for f with initial point $\lambda^{\{0\}}$, then every subsequence of $\{\lambda^{\{k\}}\}$ converges. The limit of any convergent subsequence of $\{\lambda^{\{k\}}\}$ is a strict non-negative associated gradient projection point (S-NN-GPP), as defined in Definition 4.3.2, of f.*

Proof. Our assertion that f is strict associated derivative unimodal as defined in Definition 4.4.2 implies that f has only one S-NN-GPS $S_{S-NN} \subset \Lambda$ as defined in Definition 4.3.3 at $\lambda^* \in \Lambda$. Let $\lambda^r \in S_{S-NN}$ such that $|\lambda^{\{k\}} - \lambda^r|$ is a maximum. Consider a sequence

of 1-balls $\{B(b_k, \epsilon_k)\}$ defined around $b_k = \frac{1}{2}(\lambda^{\{k\}} + \lambda^r)$ with radius of $\frac{1}{2}|\lambda^{\{k\}} - \lambda^r|$. Then every $\lambda^{\{k+1\}} \in B(b_k, \epsilon_k)$, since $\{\lambda^{\{k\}}\}$ is an associated derivative descent sequence as defined in Definition 4.3.4 and f is strict associated derivative unimodal as defined in Definition 4.4.2. Therefore, $k \to \infty$ implies $|\lambda^{\{k\}} - \lambda^r| \to 0$. It follows from the Cauchy criterion for sequences that $\{\lambda^{\{k\}}\}$ is convergent, which completes the proof of our first assertion.

Now let $\{\lambda^{\{k\}_m}\}$ be a convergent subsequence of $\{\lambda^{\{k\}}\}$ and let λ^{m*} be its limit. Suppose, contrary to the second assertion of the theorem, that λ^{m*} is not a S-NN-GPP as defined in Definition 4.3.2 of f. Since we assume that λ^{m*} is not a S-NN-GPP, and by Definition 4.3.4, there exist a $\lambda^{m*} + \delta$ for $\delta \neq 0 \in \mathbb{R}$ such that $f'^A(\lambda^{m*} + \delta) < 0$, which contradicts our assumption that λ^{m*} is the limit of the subsequence $\{\lambda^{\{k\}_m}\}$. Therefore, for λ^{m*} to be the limit of an associated derivative descent subsequence $\{\lambda^{\{k\}_m}\}$, $\lambda^{m*} \in S_{S-NN}$, which completes the proof. □

We now proceed with a proof of convergence for generalized univariate associated derivative unimodal functions when associated derivative descent sequences are considered.

Theorem 4.4.2. *Let $f : \Lambda \subseteq \mathbb{R} \to]-\infty, \infty]$ be a univariate function that is associated derivative unimodal, as defined in Definition 4.4.2, with first* associated derivative *$f'^A : \Lambda \to]-\infty, \infty[$ uniquely defined everywhere on Λ. If $\lambda^{\{0\}} \in \Lambda$ and $\{\lambda^{\{k\}}\}$ is an associated derivative descent sequence, as defined in Definition 4.3.4, for f with initial point $\lambda^{\{0\}}$, then every subsequence of $\{\lambda^{\{k\}}\}$ converges. The limit of any convergent subsequence of $\{\lambda^{\{k\}}\}$ is a generalized G-NN-GPP, as defined in Definition 4.3.1, of f.*

Proof. Our assertion that f is associated derivative unimodal as defined in Definition 4.4.2 implies that f has at least one G-NN-GPS $S_{G-NN} \in \Lambda$ as defined in Definition 4.3.3. Let $S \subset \Lambda$ be the union of G-NN-GPSs S_{G-NN}. Consider the j^{th} sequence of 1-balls $\{B(b_k, \epsilon_k)\}_j$ defined around $b_k = \frac{1}{2}(\lambda^{\{k\}} + (\lambda_j^* \in S))$ and with radius $\epsilon_k = \frac{1}{2}|\lambda^{\{k\}} - (\lambda_j^* \in S)|$. Then $\lambda^{\{k+1\}} \in B(b_k, \epsilon_k)_j$ for every sequence j since $\{\lambda^{\{k\}}\}$ is a associated derivative descent sequence as defined in Definition 4.3.4 and f is associated derivative unimodal as defined in Definition 4.4.2. Therefore $k \to \infty$ implies $|\lambda^{\{k\}} - (\lambda_j^* \in S)| \to a_j$ with a_j a constant. Since $|\lambda^{\{k\}} - (\lambda_j^* \in S)| - a_j \to 0$ for every j it follows from the Cauchy criterion for sequences that $\{\lambda^{\{k\}}\}$ is convergent, which completes the proof of our first assertion.

Now let $\{\lambda^{\{k\}_m}\}$ be a convergent subsequence of $\{\lambda^{\{k\}}\}$ and let λ^{m*} be its limit. Suppose, contrary to the second assertion of the theorem, that λ^{m*} is not a G-NN-GPP as defined in Definition 4.3.1 of f. Since we assume that λ^{m*} is not a G-NN-GPP, and by Definition 4.3.4, there exist a $\lambda^{m*} + \delta$ for $\delta \neq 0 \in \mathbb{R}$ such that $f'^A(\lambda^{m*} + \delta) < 0$ which contradicts our assumption that λ^{m*} is the limit of the subsequence $\{\lambda^{\{k\}_m}\}$. Therefore, for λ^{m*} to be the limit of an associated derivative descent subsequence (see Definition 4.3.4) $\{\lambda^{\{k\}_m}\}$, $\lambda^{m*} \in S$, which completes the proof. □

Now that we have concluded our proofs of (strictly) associated derivative unimodal univariate functions, we present a proof of convergence for univariate associated derivative coercive functions that have at least one S-NN-GPS.

Theorem 4.4.3. *Let $f : \Lambda \subseteq \mathbb{R} \to]-\infty, \infty]$ be a univariate associated derivative coercive function, as defined in Definition 4.4.1, with first* associated derivative *$f'^{A} : \Lambda \to]-\infty, \infty[$ uniquely defined everywhere on Λ. If $\lambda^{\{0\}} \in \Lambda$ and $\{\lambda^{\{k\}}\}$ is an associated derivative descent sequence, as defined in Definition 4.3.4, for f with initial point $\lambda^{\{0\}}$, then there exists at least one convergent subsequence of $\{\lambda^{\{k\}}\}$. The limit of any convergent subsequence of $\{\lambda^{\{k\}}\}$ is a S-NN-GPP of f.*

Proof. Since we only consider associated derivative descent sequences $\{\lambda^{\{k\}}\}$ our assertion that f is associated derivative coercive implies the closed interval $[a, b] \subset \Lambda$. The sequence $\{\lambda^{\{k\}}\}$ is bounded which follows from our premise of f. It follows from the Weierstrass-Bolzano theorem that in a closed interval $[a, b]$, every sequence has a subsequence that converges to a point in the interval [8].

Now let $\{\lambda^{\{k\}_m}\}$ be a convergent subsequence of $\{\lambda^{\{k\}}\}$ and let $\lambda^{m*} \in \Lambda$ be its limit. Suppose, contrary to the second assertion of the theorem, that λ^{m*} is not a S-NN-GPP of f. Since we assume that λ^{m*} is not a S-NN-GPP, and by Definition 4.3.4, there exist a $\lambda^{m*} + \delta$ for $\delta \neq 0 \in \mathbb{R}$ such that $f'^{A}(\lambda^{m*} + \delta) < 0$, which contradicts our assumption that λ^{m*} is the limit of the subsequence $\{\lambda^{\{k\}_m}\}$. Therefore, for λ^{m*} to be the limit of an associated derivative descent sequence (see Definition 4.3.4) $\{\lambda^{\{k\}_m}\}$, $\lambda^{m*} \in S_{S-NN}$ with $S_{S-NN} \subset \Lambda$ which completes the proof. □

4.4.2 Multivariate functions

We begin our proof of convergence of associated derivative descent sequences for multivariate functions with C^1 continuous convex functions [41], whereupon we present proofs of convergence for broader classes of functions.

Theorem 4.4.4. *Suppose $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is a C^1 continuous convex function with $\boldsymbol{x} \in X$. If $\boldsymbol{x}^{\{0\}} \in X$ and $\{\boldsymbol{x}^{\{k\}}\}$ is an associated derivative descent sequence, as defined in Definition 4.3.4, for f with initial point $\boldsymbol{x}^{\{0\}}$, then every subsequence of $\{\boldsymbol{x}^{\{k\}}\}$ converges. The limit of any convergent sequence of $\{\boldsymbol{x}^{\{k\}}\}$ is a S-NN-GPP as defined in Definition 4.3.2 of f.*

Proof. Our assertion that f is convex and C^1 continuous ensures that f has a single global minimizer $\boldsymbol{x}_g^* \in X$. Also, by Definition 4.3.4 and the continuity of the first partial derivatives, we see that $\{f(\boldsymbol{x}^{\{k\}})\}$ is a decreasing sequence that is bounded below by $f(\boldsymbol{x}_g^*)$. It follows that $\{\boldsymbol{x}^{\{k\}}\}$ is a bounded sequence since f is convex. The Bolzano-Weierstrass theorem implies that $\{\boldsymbol{x}^{\{k\}}\}$ has at least one convergent subsequence, which completes the proof of our first assertion [41].

Now let $\{\boldsymbol{x}^{\{k\}_m}\}$ be a convergent subsequence of $\{\boldsymbol{x}^{\{k\}}\}$ and let $\boldsymbol{x}^{m*} \in X$ be its limit. Suppose, contrary to the second assertion of the theorem, that $\boldsymbol{x}^{m*}$ is not a S-NN-GPP as defined in Definition 4.3.2 of f which from our continuity assumption implies $\nabla_A f(\boldsymbol{x}^{m*}) \neq \boldsymbol{0}$, which in turn implies that there exists a descent direction $\boldsymbol{u}^{m*}$ at $\boldsymbol{x}^{m*}$, such that $\boldsymbol{u}^{m*} \neq \boldsymbol{0}$.

Since $\{\boldsymbol{x}^{\{k\}_m}\}$ is an associated derivative descent sequence as defined in Definition 4.3.4 of which the limit $\boldsymbol{x}^{m*}$ is by assumption not a S-NN-GPP i.e.

$$-\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{m*}) \nabla_A f(\boldsymbol{x}^{m*}) < 0.$$

It follows from the continuity assumptions that there exists a small $\lambda > 0 \in R$ such that $-\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{m*} + \lambda \boldsymbol{u}^{m*}) \nabla_A f(\boldsymbol{x}^{m*}) < 0$ which contradicts our assumption that $\boldsymbol{x}^{m*}$ is the limit of the sequence $\{\boldsymbol{x}^{\{km\}}\}$. Therefore, for $\boldsymbol{x}^{*m}$ to be the limit of an associated derivative descent sequence $\{\boldsymbol{x}^{\{km\}}\}$, $\nabla_A f(\boldsymbol{x}^{m*}) = \boldsymbol{0}$, which in turn implies $\boldsymbol{u}^{m*} = \boldsymbol{0}$. The limit $\boldsymbol{x}^{*m}$ of an associated derivative descent sequence as defined in Definition 4.3.4, is therefore a S-NN-GPP as defined in Definition 4.3.2, which completes the proof. □

Before we proceed to present a proof of convergence for C^1 continuous associated derivative coercive functions, we show that if a function is associated derivative coercive and C^1 continuous it has at least one global minimizer.

Proposition 4.4.5. *Suppose $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is a C^1 continuous associated derivative coercive function as defined in Definition 4.4.1 with $\boldsymbol{x} \in X$, then f has at least one S-NN-GPP as defined in Definition 4.3.2.*

Proof. Let $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3 \in \mathbb{R}^n$. Since f is associated derivative coercive as defined in Definition 4.4.1, there exists by definition a number R_M such that for every $\{\boldsymbol{x}^2 : \|\boldsymbol{x}^2\| > R_M\}$, and every $\{\boldsymbol{x}^1 : \|\boldsymbol{x}^1\| < R_M\}$, the following holds: $\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^2)(\boldsymbol{x}^2 - \boldsymbol{x}^1) > 0$, for non perpendicular $\nabla_A f(\boldsymbol{x}^2)$ and $(\boldsymbol{x}^2 - \boldsymbol{x}^1)$. In addition, there exists $\{\boldsymbol{x}^3 : \|\boldsymbol{x}^3\| < R_M\}$, such that $\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^3)(\boldsymbol{x}^3 - \boldsymbol{x}^1) > 0$. Therefore, the set $\{\boldsymbol{x} : \|\boldsymbol{x}\| < R_M\}$ is closed and bounded, which by the continuity assumption implies that $f(\boldsymbol{x})$ assumes a minimum value on $\{\boldsymbol{x} : \|\boldsymbol{x}\| < R_M\}$ at a point $\boldsymbol{x}_g^* \in X$. From the continuity assumption of the first *partial associated derivatives*, it follows that $\nabla_A f(\boldsymbol{x}_g^*) = \boldsymbol{0}$ [41]. It therefore follows from the continuity assumptions that Definition 4.3.2 holds at $\boldsymbol{x}_g^*$. □

Theorem 4.4.6. *Suppose $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is a C^1 continuous associated derivative coercive function, as defined in Definition 4.4.1, with $\boldsymbol{x} \in X$. If $\boldsymbol{x}^{\{0\}} \in X$, and $\{\boldsymbol{x}^{\{k\}}\}$ is a conservative associated derivative descent sequence, as defined in Definition 4.3.5, for f with initial point $\boldsymbol{x}^{\{0\}}$, then some subsequence of $\{\boldsymbol{x}^{\{k\}}\}$ converges. The limit of any convergent sequence of $\{\boldsymbol{x}^{\{k\}}\}$ is a G-NN-GPP, as defined in Definition 4.3.1, of f.*

Proof. Our assertion that f is continuous and associated derivative coercive ensures that f has a global minimizer $\boldsymbol{x}_g^* \in X$. Also, by the definition of a conservative associated derivative descent sequence and the continuity of the first *partial associated derivatives*, we see that $\{f(\boldsymbol{x}^{\{k\}})\}$ is a decreasing sequence that is bounded below by $f(\boldsymbol{x}_g^*)$. Note that we require *conservative associated derivative* descent sequences, since *derivative* descent sequence is not sufficient to guarantee convergence as it may result in oscillatory behavior for $n > 1$. The remainder of the proof is similar to the proof of Theorem 4.4.4. □

We now proceed to functions that are either C^0 continuous or discontinuous, but for which the function values and *associated gradient* field are uniquely defined everywhere. We present classes of C^0 continuous or discontinuous functions for which convergence is guaranteed, since associated derivative descent sequences may not converge to NN-GPP when all C^0 continuous or discontinuous functions are considered, as is evident from the following example.

Consider the linear programming problem of finding the intersection between two intersecting planes. Since the *associated gradient* on each plane is constant, a steepest descent sequence that terminates at the intersection of the two planes is an example of a sequence that converges to some point that is not a NN-GPP.

Hence, we now present classes of well-posed discontinuous functions for which convergence is guaranteed.

Definition 4.4.4. We consider the (resp. generalized / strict) gradient-only optimization problem to be well-posed (resp. convex / unimodal) associated derivative when

1. the *associated gradient* field is everywhere uniquely defined,
2. the problem is associated derivative coercive as defined in Definition 4.4.1,
3. there exits one and only one (resp. G/S)-NN-GPS (resp. S_{G-NN} / S_{S-NN}) as defined in Definition 4.3.3, and
4. when every associated derivative descent sequence as defined in Definition 4.3.4 has at least one converging subsequence to a point in (resp. S_{G-NN} / S_{S-NN}).

We now present a class of well-posed associated derivative coercive functions; this includes multimodal functions.

Definition 4.4.5. We consider the gradient-only optimization problem to be (resp. proper / generalized) well-posed associated derivative coercive when

1. the *associated gradient* field is everywhere uniquely defined,
2. the problem is associated derivative coercive as defined in Definition 4.4.1,

3. there exits at least one (resp. G/S)-NN-GPS (resp. S_{G-NN} / S_{S-NN}) as defined in Definition 4.3.3, and

4. when every conservative associated derivative descent sequence as defined in Definition 4.3.5 has at least one converging subsequence to a point in (resp. S_{G-NN} / S_{S-NN}).

We note that the classes of functions defined in Definitions 4.4.4 - 4.4.5 still exclude many problems of practical significance e.g. linear programming problems. Many of these practically significant problems may be accommodated by altering Definitions 4.4.4 - 4.4.5 to hold only for specific associated derivative descent sequences.

4.5 Practical algorithmic considerations

We now consider some practical algorithmic implications of the foregoing, relying in particular on the new definitions for an associated derivative critical point presented in Definitions 4.3.1 and 4.3.2.

We aim to give a fairly general outline for modifying classical gradient based optimization algorithms to become gradient-only optimization algorithms; often this merely requires subtle modifications to conventional gradient based algorithms. We consider two classes of optimization algorithms, namely line search descent methods, and approximation methods; both are prevalent in practical optimization.

4.5.1 Line search descent methods

Line search methods are generally present in first order methods (e.g. steepest descent and conjugate gradient methods), and second order methods (e.g. in general the modified Newton methods e.g. Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS)) [55]. In any event, for a given iteration k, the current position is given by $\boldsymbol{x}^{\{k-1\}}$, $k = 1, 2, 3, \ldots$ and search direction $\boldsymbol{u}^{\{k\}}$ at $\boldsymbol{x}^{\{k-1\}}$. In general, line search methods use function values along the search direction $\boldsymbol{u}^{\{k\}}$. In formulating a rudimentary gradient-only algorithm, the line search simply needs to be modified to only consider the *associated directional derivative* along the search direction $\boldsymbol{u}^{\{k\}}$. Let us therefore consider line search bracketing strategies (of which Fibonacci and golden ratio searches are examples) in the following.

Function-value based bracketing strategies require a minimum of three points to bracket a minimum of $F(\lambda) = f(\boldsymbol{x}^{\{k-1\}} + \lambda \boldsymbol{u}^{\{k\}})$ along $\boldsymbol{u}^{\{k\}}$. Consequently, three points from the sequence $[w(l-1), w(l), w(l+1)]$, $\quad l = 1, 2, \ldots, l_{max}$, are used with $w(j) = \gamma j$. The line search iterations l are incremented until either a minimum is located or the maximum number of line search iterations l_{max} are reached.

Once an interval that contains the minimum is located, a minimum of four points i.e. three intervals are required to refine the minimum. The aim of function-value based bracketing strategies is to find $\lambda^{\{k\}}$ such that for $\xi \in \mathbb{R}$ and $\xi > 0$:

$$F(\lambda^{\{k\}} + \xi) = f\left(\boldsymbol{x}^{\{k-1\}} + (\lambda^{\{k\}} + \xi)\boldsymbol{u}^{\{k\}}\right) > F(\lambda^{\{k\}}),$$

and

$$F(\lambda^{\{k\}} - \xi) = f\left(\boldsymbol{x}^{\{k-1\}} + (\lambda^{\{k\}} - \xi)\boldsymbol{u}^{\{k\}}\right) > F(\lambda^{\{k\}}). \tag{4.7}$$

Line search bracketing strategies are easily modified to use only *associated gradient* information, e.g. see Bazaraa *et al.* [4], as we will now illustrate.

Modifying bracketing strategies to require only *associated gradient* information requires a minimum of two points to bracket a sign change in the *associated directional derivative* along $\boldsymbol{u}^{\{k\}}$ from negative to positive. Therefore, two points from the sequence $[w(l-1), w(l)], \quad l = 1, 2, \ldots, l_{max}$, are used with $w(j) = \gamma j$. The line search iterations l are incremented until either a sign change in the *associated directional derivative* $F'^{A}(\lambda)$ is located or the maximum number of line search iterations l_{max} are reached.

Once an interval is located that contains a sign change, a minimum of three points i.e. two intervals are required to refine the location of the sign change. The aim of gradient-only bracketing strategies is to locate $\lambda^{\{k\}}$ such that:

$$F'^{A}(\lambda^{\{k\}} + \xi) = \left[\nabla_A{}^{\mathrm{T}} f\left(\boldsymbol{x}^{\{k-1\}} + \boldsymbol{u}^{\{k\}}(\lambda^{\{k\}} + \xi)\right)\right] \boldsymbol{u}^{\{k\}} > 0 \tag{4.8}$$

and that

$$F'^{A}(\lambda^{\{k\}} - \xi) = \left[\nabla_A{}^{\mathrm{T}} f\left(\boldsymbol{x}^{\{k-1\}} + \boldsymbol{u}^{\{k\}}(\lambda^{\{k\}} - \xi)\right)\right] \boldsymbol{u}^{\{k\}} < 0. \tag{4.9}$$

Note that we merely locate the point at which the *associated directional derivative* changes from negative to positive. The requirement for the *associated directional derivative* to equal zero, is therefore relaxed. This is particularly important when considering discontinuous functions. However, for smooth functions, the sign change from negative to positive of course occurs at the point where the *associated directional derivative* is zero. In addition, inflection points are handled appropriately, as no sign change in the *associated directional derivative* occurs over an inflection point.

Algorithmic implementation

We now consider the algorithmic implementation of the second-order line search BFGS method for unconstrained minimization. Given an initial point $\boldsymbol{x}^{\{0\}}$, the BFGS implementation proceeds as follows:

1. **Initialization:** Select real constants $\epsilon > 0$, $\xi > 0$ and $\gamma > 0$. Select integer constants k_{max} and l_{max}. Set $\boldsymbol{G}^{\{0\}} = \boldsymbol{I}$. Set $k := 0$ and $l := 0$.

2. **Gradient evaluation:** Compute $\nabla_A f(\boldsymbol{x}^{\{k\}})$.

3. **Update the search direction** $\boldsymbol{u}^{\{k+1\}} = -\boldsymbol{G}^{\{k\}}\nabla_A f(\boldsymbol{x}^{\{k\}})$.

4. **Initiate an inner loop to conduct line search:** Find $\lambda^{\{k+1\}}$ using the line search strategy described in Section 4.5.1 by either (4.7) for classical or (4.9) for gradient only.

5. **Test for re-initialization of** $\boldsymbol{G}^{\{k\}}$**:** if $k \mod n = 0$ then $\boldsymbol{G}^{\{k\}} = \boldsymbol{I}$ else

$$\boldsymbol{G}^{\{k\}} = \boldsymbol{G}^{\{k-1\}} + \left[1 + \frac{(\boldsymbol{y}^{\{k\}})^{\mathrm{T}}\boldsymbol{G}^{\{k-1\}}\boldsymbol{y}^{\{k\}}}{(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}\boldsymbol{y}^{\{k\}}}\right]\left[\frac{\boldsymbol{v}^{\{k\}}(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}}{(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}\boldsymbol{y}^{\{k\}}}\right] - \left[\frac{\boldsymbol{v}^{\{k\}}(\boldsymbol{y}^{\{k\}})^{\mathrm{T}}\boldsymbol{G}^{\{k-1\}} + \boldsymbol{G}^{\{k-1\}}\boldsymbol{y}^{\{k\}}(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}}{(\boldsymbol{v}^{\{k\}})^{\mathrm{T}}\boldsymbol{y}^{\{k\}}}\right],$$

   with $\boldsymbol{v}^{\{k\}} = \lambda^{\{k\}}\boldsymbol{u}^{\{k\}}$ and $\boldsymbol{y}^{\{k\}} = \left(\nabla_A f(\boldsymbol{x}^{\{k\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}})\right)$.

6. **Move to the new iterate:** Set $\boldsymbol{x}^{\{k+1\}} := \boldsymbol{x}^{\{k\}} + \lambda^{\{k+1\}}\boldsymbol{u}^{\{k+1\}}$.

7. **Convergence test:** if $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \leq \epsilon$ OR $k = k_{\max}$, stop.

8. **Initiate an additional outer loop:** Set $k := k + 1$ and goto Step 2.

4.5.2 Approximation methods

Approximation methods can also be formulated using only *associated gradient* information, e.g. see Groenwold *et al.* [22].

Let us consider approximation functions $\tilde{f}$ that use the *second order* Taylor series expansion of a function f around some current iterate $\boldsymbol{x}^{\{k\}}$, given by

$$\tilde{f}^{\{k\}}(\boldsymbol{x}) = f(\boldsymbol{x}^{\{k\}}) + {\nabla_A}^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}\boldsymbol{H}^{\{k\}}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}),\ k = 0, 1, 2, \ldots \quad (4.10)$$

where superscript k represents an iteration number, $\tilde{f}$ the second order Taylor series approximation to f, ∇_A the *associated gradient* operator and $\boldsymbol{H}^{\{k\}}$ the Hessian. $f(\boldsymbol{x}^{\{k\}})$ and $\nabla_A f(\boldsymbol{x}^{\{k\}})$ respectively represent the function value and *associated gradient* vector at the current iterate $\boldsymbol{x}^{\{k\}}$. Generally speaking, approximation methods use only function value information in constructing $\boldsymbol{H}^{\{k\}}$ (due to the excessive computational effort associated with evaluating and storing $\boldsymbol{H}^{\{k\}}$ in the first place).

Consider for example a diagonal spherical quadratic approximation, with $\boldsymbol{H}^{\{k\}} = c^{\{k\}}\boldsymbol{I}$. The unknown $c^{\{k\}}$ can be obtained by enforcing $\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) = f(\boldsymbol{x}^{\{k-1\}})$, which

results in

$$f(\boldsymbol{x}^{\{k-1\}}) = f(\boldsymbol{x}^{\{k\}}) + \nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) + \frac{c^{\{k\}}}{2}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}), \quad (4.11)$$

e.g. see Snyman and Hay [56]. The scalar $c^{\{k\}}$ is then obtained as

$$c^{\{k\}} = 2\frac{f(\boldsymbol{x}^{\{k-1\}}) - f(\boldsymbol{x}^{\{k\}})}{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})} - 2\frac{\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})}{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})}. \quad (4.12)$$

Approximations solely based on gradient information may be constructed by taking the derivative of (4.10), which gives

$$\nabla\tilde{f}^{\{k\}}(\boldsymbol{x}) = \nabla_A f(\boldsymbol{x}^{\{k\}}) + \boldsymbol{H}^{\{k\}}(\boldsymbol{x} - \boldsymbol{x}^{\{k\}}), \quad k = 0, 1, 2, \ldots \quad (4.13)$$

Note that at $\boldsymbol{x} = \boldsymbol{x}^{\{k\}}$, the *associated gradient* of the function $f(\boldsymbol{x})$ exactly match the gradient of the approximation function $\tilde{f}(\boldsymbol{x})$. Notationally we write the gradient instead of *associated gradient* of the approximation function to emphasize the differentiability of the approximation function.The Hessian $\boldsymbol{H}^{\{k\}}$ of the approximation $\tilde{f}$ is chosen to match some additional condition. Let us again consider a spherical quadratic approximation, with $\boldsymbol{H}^{\{k\}} = c^{\{k\}}\boldsymbol{I}$. Then, $c^{\{k\}}$ may be obtained by matching the gradient vectors at $\boldsymbol{x}^{\{k-1\}}$. Since only a single free parameter $c^{\{k\}}$ is available, the n components of the respective gradient vectors can (for example) be matched in a least square sense.

The least squares error is given by

$$E^{\{k\}} = (\nabla\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}}))^{\mathrm{T}}(\nabla\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}})). \quad (4.14)$$

After substitution of $\nabla_A\tilde{f}^{\{k\}}(\boldsymbol{x}^{\{k-1\}}) = \nabla_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})$, we have

$$E^{\{k\}} = (\nabla_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}}))^{\mathrm{T}} (\nabla_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}})). \quad (4.15)$$

Minimization of the least squares error $E^{\{k\}}$ w.r.t. $c^{\{k\}}$ then gives

$$\frac{dE^{\{k\}}}{dc^{\{k\}}} = (\nabla_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}}))^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) + (\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\nabla_A f(\boldsymbol{x}^{\{k\}}) + c^{\{k\}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) - \nabla_A f(\boldsymbol{x}^{\{k-1\}})) = 0, \quad (4.16)$$

hence

$$c^{\{k\}} = \frac{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\nabla_A f(\boldsymbol{x}^{\{k-1\}}) - \nabla_A f(\boldsymbol{x}^{\{k\}}))}{(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})}. \tag{4.17}$$

If the approximation is required to be strictly convex, we can enforce $c^{\{k\}} = \max(\beta, c^{\{k\}})$, with $\beta > 0$ small and prescribed.

Since the sequential *approximate* subproblems are smooth, they may be solved analytically; the minimizer of subproblem k follows from setting (4.13) equal to $\mathbf{0}$ [52], to give

$$\boldsymbol{x}^{\{k*\}} = \boldsymbol{x}^{\{k\}} - \frac{\nabla_A f(\boldsymbol{x}^{\{k\}})}{c^{\{k\}}}. \tag{4.18}$$

4.5.3 Conservative approximations

Global convergence of sequential approximation methods may for example be affected through the notion of conservatism. Classical conservatism is based solely on function values, for which Svanberg [58] demonstrated that an approximation sequence $k = 1, 2, \cdots$ will terminate at the global minimizer $\boldsymbol{x}^* \leftrightarrow f^*$, if each k^{th} approximation $\tilde{f}(\boldsymbol{x}^{\{k*\}})$ is conservative, i.e. if

$$\tilde{f}(\boldsymbol{x}^{\{k*\}}) \geq f(\boldsymbol{x}^{\{k*\}}) \; \forall \; k. \tag{4.19}$$

Conservatism may also be affected using only *associated gradient* information. At iterate $\boldsymbol{x}^{\{k*\}}$, the update is given by $\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}$, and conservatism is affected if the projection of the *associated gradient* $\nabla_A f(\boldsymbol{x}^{\{k*\}})$ of the actual function $f(\boldsymbol{x})$ onto the update direction $\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}$ is negative. For univariate functions, an update is conservative if it is an associated derivative descent update step (see Definition 4.3.4). For multivariate functions an update is conservative if it is a conservative associated derivative descent update step (see Definition 4.3.5), i.e. if

$$\nabla_A{}^{\mathrm{T}} f(\boldsymbol{x}^{\{k*\}})(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}) < 0. \tag{4.20}$$

Hence, enforcement of the conditions given by Definition 4.3.4 or 4.3.5 suffice to ensure a sequence of derivative descent sequences for which proofs of convergence are offered in Section 4.4. To allow for update steps that are computable we employ a trust region strategy where we limit $\|\boldsymbol{x}^* - \boldsymbol{x}^{\{k\}}\| \leq \gamma$.

Algorithmic implementation

Given an initial point $\boldsymbol{x}^{\{0\}}$, a {gradient-only}/classical conservative algorithm based on convex separable spherical quadratic approximations (SSA) for unconstrained minimization proceeds as follows:

1. **Initialization:** Select real constants $\epsilon > 0$, $\alpha > 1$ and initial curvature $c^{\{0\}} > 0$. Set $k := 0$, $l := 0$.

2. **Gradient evaluation:** Compute $\{\nabla_A f(\boldsymbol{x}^{\{k\}})\}/f(\boldsymbol{x}^{\{k\}})$ and $\nabla_A f(\boldsymbol{x}^{\{k\}})$.

3. **Approximate optimization:** Construct local approximate subproblem $\{(4.13)\}/(4.10)$ at $\boldsymbol{x}^{\{k\}}$, using $\{(4.17)\}/(4.12)$ unless inside an inner loop then use $\boldsymbol{c}^{\{k\}}$ as calculated in Step 6(b). Solve this subproblem analytically, to arrive at $\boldsymbol{x}^{\{k*\}}$.

4. **Evaluation:** Compute $\{\nabla_A f(\boldsymbol{x}^{\{k*\}})\}/f(\boldsymbol{x}^{\{k*\}})$.

5. **Test if $\boldsymbol{x}^{\{k*\}}$ is acceptable:** if $\{(4.20)\}/(4.19)$ is satisfied, goto Step 7.

6. **Initiate an inner loop to effect conservatism:**

   (a) Set $l := l + 1$.

   (b) Set $c^{\{k\}} := \alpha c^{\{k\}}$.

   (c) Goto Step 3.

7. **Move to the new iterate:** Set $\boldsymbol{x}^{\{k+1\}} := \boldsymbol{x}^{\{k*\}}$.

8. **Convergence test:** if $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \leq \epsilon$, OR $k = k_{max}$, stop.

9. **Initiate an additional outer loop:** Set $k := k + 1$ and goto Step 2.

4.5.4 Termination criteria

Termination criteria also need some consideration: if the function values and *associated gradients* of an objective or cost function contain step discontinuities, these quantities may not provide robust termination information. Accordingly, we only advocate the robust termination criterion

$$\|\Delta\boldsymbol{x}^{\{k+1\}}\| = \|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| < \epsilon, \tag{4.21}$$

with ϵ small, positive and prescribed. (A maximum number of iterations may of course also be prescribed, but this is not robust.)

4.6 Mathematical programming vs. gradient-only optimization

We now briefly reflect on some differences between gradient-only optimization and classical 'mathematical programming'. Consider the step discontinuities depicted in Figure 4.5.

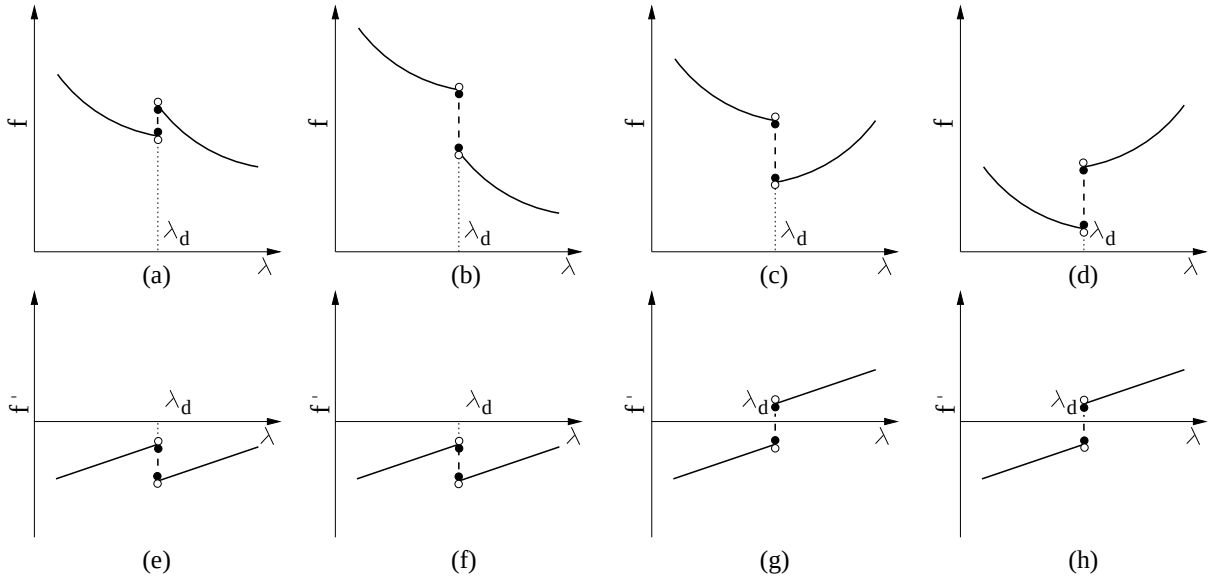


Figure 4.5: Plots depicting (a)-(d) the function values, and (e)-(h) the corresponding *associated derivatives* of four instances of step discontinuous univariate functions.

In classical mathematical programming, the inconsistent step discontinuity depicted in Figure 4.5(a) result in a local minimum, whereas the function with the consistent step discontinuity depicted in Figure 4.5(b) is monotonically decreasing. The step discontinuities depicted in Figures 4.5(c)-(d) again result in local minima.

In gradient-only optimization, the inconsistent step discontinuity in Figure 4.5(a) is associated derivative negative, as is the consistent step discontinuity depicted in Figure 4.5(b). The step discontinuities depicted in Figures 4.5(c)-(d) represent non-negative gradient projection points (S-NN-GPPs) as shown in the *associated* derivative of Figures 4.5(c)-(d).

Consider the objective functions depicted in Figure 4.6 (b). Clearly, classical optimization approaches may get stuck in local minima caused by inconsistent step discontinuities, whereas gradient-only optimization approaches will not. Hence, gradient-only optimization allows for a robust strategy to avoid inconsistent step discontinuities when the minimizer x^* of an objective function coincides with a strict non-negative gradient projection point (S-NN-GPP) x_g^* as shown in Figure 4.6 (b).

However, gradient-only approaches will ignore a global minimizer x^* of an objective function that occurs over an inconsistent step discontinuity as depicted in Figure 4.6 (a) and converge to a S-NN-GPP x_g^*. Hence, whether function-value based or gradient-only based criteria is to be used will depend on which one best describes or approximates the solution of an optimization problem.

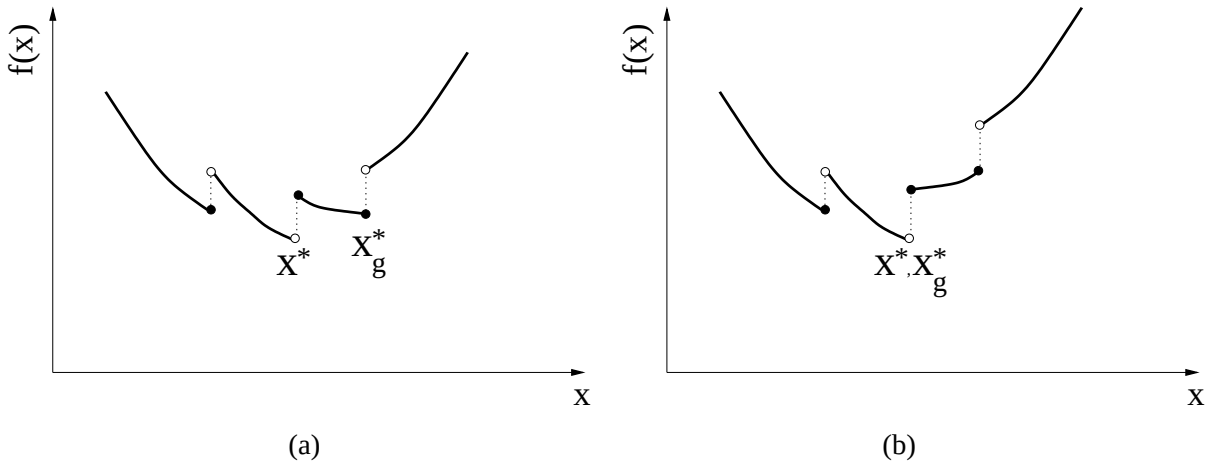


Figure 4.6: Plots depicting a step discontinuous objective function with a (a) distinct minimizer x^* and strict non-negative gradient projection point (S-NN-GPP) x_g^* and (b) coinciding minimizer x^* and S-NN-GPP x_g^*.

4.7 Numerical study

We start our numerical study with a practical shape optimization problem using a remeshing strategy that results in a discontinuous objective function. We then proceed with a set of discontinuous test functions aimed to "mimic" non-physical discontinuities in functions. The advantage of introducing a set of test problems is that they are easily implemented which allows for focussed research on algorithm development and testing, without requiring access to a variable discretization PDE solver. The disadvantage of test problems is that only part of the complexity of PDE based objective functions is captured.

The algorithmic settings used in the numerical study are presented in Table 4.1 for the algorithms outlined in Sections 4.5.1 and 4.5.3. The choice of $\gamma = 0.1$ is deliberate as we aim to "mimic" a locally exact line search at the cost of computational efficiency and the results should be interpreted in view of this. The aim is to highlight the differences between function value and gradient-only based line search strategies. It is evident that the probability of getting trapped locally using function-value based line search strategies is reduced when γ is increased or when some interpolation strategy is used in the line search e.g. Powell's method [55]. The latter being evident from the approximation results; we however note that although many non-physical local minima may be avoided using either of the two strategies, neither is robust, and the algorithms may still get trapped in non-physical local minima.

Table 4.1: Algorithmic settings used in the numerical experiments.

ϵ	γ	ζ	α	k_{max}	l_{max}
10^{-5}	0.1	10^{-6}	2	3000	3000

Table 4.2: Tabulated results obtained for the unconstrained Michell-like structure.

Algorithm	$f(\boldsymbol{x}^{\{N_k\}})$	$\|\nabla_A f(\boldsymbol{x}^{\{N_k\}})\|$	$\|\Delta\boldsymbol{x}^{\{N_k\}}\|$	N_k	N_l
BFGS(f)	7.293E-01	3.163E-02	0.000E+00	5	137
BFGS(g)	5.213E-01	3.118E-03	0.000E+00	38	816
SSA(f)	5.805E-01	1.719E-02	7.896E-06	23	10
SSA(g)	5.285E-01	3.438E-03	3.249E-06	103	7

4.7.1 Results

4.7.2 Shape optimization

We now consider the isotropic shape optimization problem outlined in Section 4.1.2. In addition to the algorithm settings given in Table 4.1 we also limit the maximum step size of each algorithm to 2. The results for the BFGS(f), BFGS(g), SSA(f) and SSA(g) algorithms are summarized in Table 4.2 with the respective final designs depicted in Figures 4.7 (a)-(d). Recall that the (f) postfix indicates classical function-value based algorithms, whereas the (g) postfix indicates gradient-only optimization algorithms. Table 4.2 presents the function value $f(\boldsymbol{x}^{\{N_k\}})$, associated gradient norm $\|\nabla_A f(\boldsymbol{x}^{\{N_k\}})\|$, convergence tolerance $\|\Delta\boldsymbol{x}^{\{N_k\}}\|$, number of outer iterations N_k as well as the number of inner iterations N_l.

Consider BFGS(f) which converged after 5 outer iterations N_k and, evidently got trapped in a local minimum due to a numerically induced step discontinuity. The premature final design is apparent from Figure 4.7 (a).

Similarly, SSA(f) converged after 23 outer iterations N_k also after getting trapped in a step discontinuous minimum. The behaviour of the cost function around the converged solution of SSA(f) is depicted in Figure 4.2 of Section 4.1.2. Significant improvements are evident by comparing Figure 4.7 (c) to Figure 4.7 (a), although noticeable improvements could still be made. Clearly, conservative approximation methods are able to overcome *some* step discontinuities and of course even more so when conservatism is relaxed.

Conversely, BFGS(g) and SSA(g) were able to optimize the Michell structure without getting trapped in numerically induced step discontinuities. Consider the similar designs depicted in Figures 4.7 (b) and (d). It is clear that BFGS(g) and SSA(g) improved notably on the designs obtained with BFGS(f) and SSA(f).

We further present for each algorithm their respective histories w.r.t. function value $f(\boldsymbol{x}^{\{k\}})$, associated gradient norm $\|\nabla_A f(\boldsymbol{x}^{\{k\}})\|$ and convergence tolerance $\|\Delta\boldsymbol{x}^{\{k\}}\|$. The

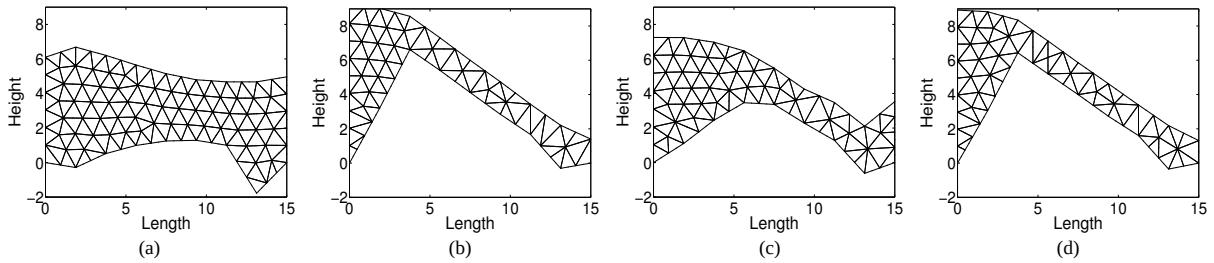


Figure 4.7: Michell-like structure: converged designs obtained with (a) BFGS(f), (b) BFGS(g), (c) SSA(f), and (d) SSA(g).

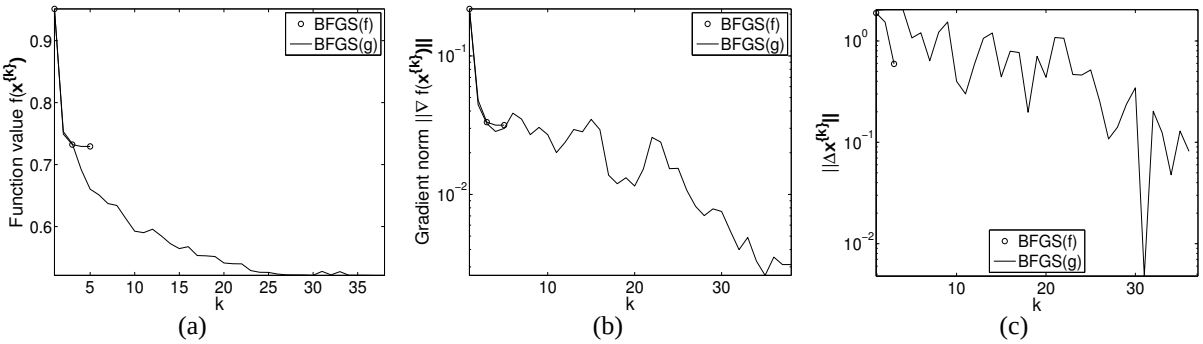


Figure 4.8: Michell-like structure: BFGS(f) and BFGS(g) algorithms convergence history plot of the (a) function value $f(x^{\{k\}})$ and (b) associated gradient norm $\|\nabla_A f(\boldsymbol{x}^{\{k\}})\|$ (c) and convergence tolerance $\|\Delta \boldsymbol{x}^{\{k\}}\|$.

respective histories for the BFGS algorithms are depicted in Figures 4.8 (a)-(c) and for the SSA algorithms in Figures 4.9 (a)-(c).

Monotonic function value decrease for both BFGS(f) and SSA(f) is clearly depicted in respectively Figure 4.8(a) and Figure 4.9(a) with the respective associated gradient norms depicted in Figure 4.8(b) and Figure 4.9(b). The convergence histories are depicted in Figure 4.8(c) and Figure 4.9(c).

Conversely, non-monotonic function value decrease for both BFGS(g) and SSA(g) is evident in Figure 4.8(a) and Figure 4.9(a) with the respective associated gradient norms

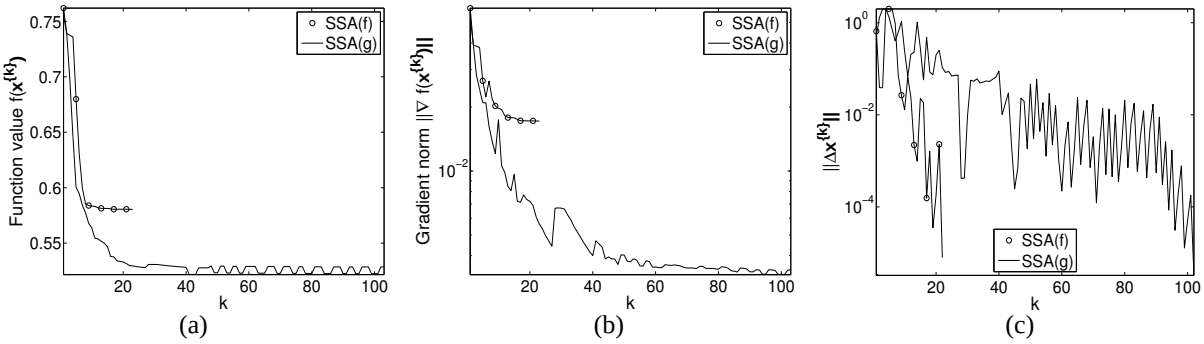


Figure 4.9: Michell-like structure: SSA(f) and SSA(g) algorithms convergence history plot of the (a) function value $f(x^{\{k\}})$ and (b) associated gradient norm $\|\nabla_A f(\boldsymbol{x}^{\{k\}})\|$ (c) and convergence tolerance $\|\Delta \boldsymbol{x}^{\{k\}}\|$.

depicted in Figure 4.8(b) and Figure 4.9(b). The convergence histories are depicted in Figure 4.8(c) and Figure 4.9(c).

4.7.3 Analytical set of test problems

We now present a set of five analytical step discontinuous test problems in order to further illustrate the advantages of gradient-only optimization.

Rosenbrock step discontinuous function f_1 is piecewise defined as follows:

$$f_1(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{n} \frac{1}{1.1}\left(100\left(x(2i) - x^2(2i-1)\right)^2 + (1 - x(2i-1))^2\right), \\ \qquad \text{if } 0 \leq \sin(2\|\boldsymbol{x}\|) < \frac{2}{3}. \\ \sum_{i=1}^{n} 1.1\left(100\left(x(2i) - x^2(2i-1)\right)^2 + (1 - x(2i-1))^2\right), \\ \qquad \text{if } -\frac{2}{3} \leq \sin(2\|\boldsymbol{x}\|) < 0. \\ \sum_{i=1}^{n} \left(100\left(x(2i) - x^2(2i-1)\right)^2 + (1 - x(2i-1))^2\right), \\ \qquad \text{if } -\frac{2}{3} > \sin(2\|\boldsymbol{x}\|) \geq \frac{2}{3}. \end{cases} \tag{4.22}$$

Quadric step discontinuous function f_2 is piecewise defined as follows:

$$f_2(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x(j)\right)^2, & \text{if } \sin(8\|\boldsymbol{x}\|) > 0.5. \\ \sum_{i=1}^{n} 1.3\left(\sum_{j=1}^{i} x(j)\right)^2, & \text{if } \sin(8\|\boldsymbol{x}\|) < -0.5. \\ \sum_{i=1}^{n} \frac{1}{1.3}\left(\sum_{j=1}^{i} x(j)\right)^2, & \text{if } -0.5 \leq \sin(8\|\boldsymbol{x}\|) \leq 0.5. \end{cases} \tag{4.23}$$

Sum squares step discontinuous function f_3 is piecewise defined as follows:

$$f_3(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{n} \frac{1}{1.5} i x^2(i), & \text{if } \sin\left(\frac{1}{10}\sum_{j=1}^{n} x(j)\right) > 0.5. \\ \sum_{i=1}^{n} 1.5 i x^2(i), & \text{if } \sin\left(\frac{1}{10}\sum_{j=1}^{n} x(j)\right) < -0.5. \\ \sum_{i=1}^{n} i x^2(i) + \frac{1}{n}, & \text{if } -0.5 \leq \sin\left(\frac{1}{10}\sum_{j=1}^{n} x(j)\right) \leq 0.5. \end{cases} \tag{4.24}$$

Zakharov step discontinuous function f_4 is piecewise defined as follows:

$$f_4(\boldsymbol{x}) = \begin{cases} \frac{1}{1.5}\sum_{i=1}^{n} x^2(i) + \left(\sum_{i=1}^{n} \frac{ix^2(i)}{2}\right)^2 + \left(\sum_{i=1}^{n} \frac{ix^2(i)}{2}\right)^4, \\ \text{if } \sin(\|\boldsymbol{x}\|) > 0.5. \\ 1.5\sum_{i=1}^{n} x^2(i) + \left(\sum_{i=1}^{n} \frac{ix^2(i)}{2}\right)^2 + \left(\sum_{i=1}^{n} \frac{ix^2(i)}{2}\right)^4 + 0.5, \\ \text{if } \sin(\|\boldsymbol{x}\|) < -0.5. \\ \sum_{i=1}^{n} x^2(i) + \left(\sum_{i=1}^{n} \frac{ix^2(i)}{2}\right)^2 + \left(\sum_{i=1}^{n} \frac{ix^2(i)}{2}\right)^4 + 1, \\ \text{if } -0.5 \leq \sin(\|\boldsymbol{x}\|) \leq 0.5. \end{cases} \tag{4.25}$$

Hyper ellipsoid step discontinuous function f_5 is piecewise defined as follows:

$$f_5(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{n} \frac{1}{1.1} 2^{i-1} x^2(i) + \frac{1}{n}, & \text{if } \sin\left(2\sum_{j=1}^{n} x(j)\right) > 0.5. \\ \sum_{i=1}^{n} 1.1 \times 2^{i-1} x^2(i) + \frac{1}{n}, & \text{if } \sin\left(2\sum_{j=1}^{n} x(j)\right) < 0, \\ \sum_{i=1}^{n} 2^{i-1} x^2(i), & \text{if } 0 \leq \sin\left(2\sum_{j=1}^{n} x(j)\right) \leq 0.5. \end{cases} \tag{4.26}$$

This set of step discontinuous test problems "mimics" functions that contain non-physical discontinuities. Our aim is to overcome the discontinuities to obtain $\boldsymbol{x}_g^*$ as outlined in Definition 4.3.3. The region around the solution of f_1, f_2 and f_4 is continuous as opposed to the region around the optima of f_3 and f_5 which are discontinuous. The solution of f_1 is given by $x^*(i) = 1,\ i = 1, 2, \ldots, n$ with $f_1^* = 0$ whereas the solution of f_2 and f_4 is given by $x^*(i) = 0,\ i = 1, 2, \ldots, n$ with $f_2^* = 0$ and $f_4^* = 1$ respectively. The solution of f_3 and f_5 is at a discontinuity and is therefore defined by a derivative critical set S. The derivative critical sets for both f_3 and f_5 are defined by $x^*(i) = 0,\ i = 1, 2, \ldots, n$. The possible function values at the optima are $f_3^* = \{0, 1\}$ and $f_5^* = \{0, 1\}$ and depends on the direction from which the discontinuity is approached. The gradient field for each test function is given by the analytical gradient of each test function whereas the gradient at a discontinuous point is defined by the analytical gradient of the active equation of a test function at that point.

Results are presented for dimension $n = 10$ of the test problem set given in Section 4.7.3. The starting point of each algorithm for each problem is $x(i)^{\{0\}} = 4,\ i = 1, 2, \ldots n$.

Numerical results are presented in Table 4.3. N_k and N_l respectively represent the number of function or gradient evaluations in the outer and inner loops. We have not limited the step size of the approximation algorithms; this is normally not done in algorithms based on conservatism (although it may sometimes be beneficial).

Table 4.3: Results for the step discontinuous test problem set.

Function	Solution	BFGS(f)	BFGS(g)	SSA(f)	SSA(g)
f_1	f^*	5.969E+04	1.520E-05	3.130E+00	5.440E-04
	$\|\nabla_A f^*\|$	3.718E+04	3.333E-03	9.599E-01	2.546E-02
	$\|\boldsymbol{x}^{\{N_k*\}} - \boldsymbol{x}^*\|$	9.771E+00	9.158E-03	5.248E+00	5.497E-02
	N_k	4	269	99	757
	N_l	66	3310	133	739
f_2	f^*	4.612E+03	1.106E-08	1.044E+01	5.017E-07
	$\|\nabla_A{}^*\|$	7.918E+02	2.083E-04	1.021E+01	9.291E-04
	$\|\boldsymbol{x}^{\{N_k*\}} - \boldsymbol{x}^*\|$	1.250E+01	2.282E-04	2.683E+00	1.540E-03
	N_k	3	173	27	101
	N_l	42	2480	106	90
f_3	f^*	8.353E+00	3.443E-10	8.371E+00	2.573E-09
	$\|\nabla_A{}^*\|$	7.999E+00	3.949E-05	6.344E+00	1.031E-04
	$\|\boldsymbol{x}^{\{N_k*\}} - \boldsymbol{x}^*\|$	2.861E+00	1.843E-05	2.981E+00	5.043E-05
	N_k	5	59	25	30
	N_l	228	965	93	18
f_4	f^*	1.136E+08	1.000E+00	3.418E+01	1.000E+00
	$\|\nabla_A{}^*\|$	4.319E+07	1.240E-03	1.157E+01	1.525E-03
	$\|\boldsymbol{x}^{\{N_k*\}} - \boldsymbol{x}^*\|$	1.204E+01	7.679E-06	5.760E+00	7.624E-04
	N_k	3	5	30	130
	N_l	47	222	50	235
f_5	f^*	1.037E+04	7.162E-07	2.055E+02	1.051E-06
	$\|\nabla_A{}^*\|$	3.371E+03	2.196E-03	1.238E+02	2.961E-03
	$\|\boldsymbol{x}^{\{N_k*\}} - \boldsymbol{x}^*\|$	1.204E+01	8.454E-04	7.072E+00	1.024E-03
	N_k	3	763	39	232
	N_l	52	10597	159	222

The results presented in Table 4.3 show that gradient-only optimization algorithms are able to robustly minimize step discontinuous objective functions. In contrast, the classical function-value based optimization algorithms converged to local minima on each of the problems. It is clear from Table 4.3 that the function-value based approximation algorithm SSA(f) is able to overcome many of the non-physical local minima. However, SSA(f) does not represent a robust strategy as it still converged to a local minimum on each of the test problems.

4.8 Conclusions

We have studied the unconstrained minimization of functions containing step or jump discontinuities and for which associated gradients can be computed everywhere. Step or jump discontinuities arises during the solution of systems of (partial) differential equations, when variable spatial and temporal discretization techniques produce discontinuities that are artifacts of the approximate numerical strategies used. While discontinuous, we demonstrate that these problems may effectively be minimized if only gradient information is used. Various algorithmic options were discussed and numerical results presented for a practical shape optimization problem as well as a set of analytical test functions. We presented a mathematical framework for gradient-only optimization that includes convergence proofs for piece-wise smooth step discontinuous functions classes of functions.

The implications of our approach are that variable discretization strategies, which are so important in numerical discretization methods, may be used in combination with efficient local optimization algorithms, notwithstanding the fact that these strategies themselves introduce step discontinuities.

CHAPTER 5

Adaptive remeshing in shape optimization

As discussed in Chapter 2, Persson and Strang have previously proposed an unstructured remeshing strategy based on a truss structure analogy, which we in turn rendered quadratically convergent. Herein, we turn our quadratically convergent mesh generator into an adaptive generator, by allowing for a spatially varying ideal element length field, computed using the Zienkiewicz-Zhu error indicator. The remeshing strategy makes (semi) analytical sensitivities available for use in gradient based optimization algorithms. To circumvent difficulties associated with local minima due to remeshing, we rely on gradient-only optimization algorithms as presented in Chapters 3 and 4. Numerical results are presented for an orthotropic cantilever beam, an orthotropic Michell-like structure and a spanner design problem.

This chapter is arranged as follows. Firstly, we present an overview of adaptive mesh refinement in shape optimization in Section 5.1, followed by a description of the gradient-only shape optimization problem in Section 5.2. Thereafter we briefly outline the gradient-only optimization algorithm used in this study in Section 5.3. We then discuss the structural analysis, including the *a posteriori* error analysis and mesh refinement strategy, in Section 5.4. Our adaptive mesh generation strategy is presented in Section 5.5, followed in Section 5.6 by a sensitivity analysis. Section 5.7 contains all the numerical results, which includes a convergence study and three example problems. Some conclusions are offered in Section 5.8.

5.1 Introduction

In Chapter 2, we presented a remeshing strategy for finite element based shape optimization [66]. Recall that this remeshing strategy is based on a truss structure analogy [42], and the equilibrium position of the truss system is solved for using Newton's method. As described the method uses (semi) analytical sensitivity information, which is computed efficiently, and which makes the use of highly efficient gradient based optimization algorithms possible. However, the numerically computed objective function of the shape optimization problem is discontinuous as shown in Chapter 3. These discontinuities are due to changes in the mesh topology[1] that result from the remeshing strategy. Many of these discontinuities manifest themselves as local minima in the objective function, causing difficulties for conventional gradient-based optimization algorithms [2].

These difficulties are often accommodated by "smoothing" of the objective function. Approaches for smoothing the objective function include the construction of inherently smooth objective functions by avoiding remeshing altogether, and by using mesh movement strategies [9]. Surrogate approaches may also be used to construct smooth representations of the discontinuous objective function, and to reduce the magnitudes of the discontinuities [48]. However, mesh movement strategies are susceptible to element distortion and inversion, while surrogate methods scale poorly with problem dimension. Controlling the discretization error is computationally expensive, since multiple finite element analyses (FEAs) are usually required for each candidate shape design [16, 23, 30, 48].

Alternatively, the discontinuous objective functions may be optimized directly. Selected approaches include conventional gradient-based optimization algorithms used in combination with restart strategies, and derivative free optimization methods [7, 11, 14]. These strategies are usually also computationally expensive due to the high number of required iterations, and/or poor scaling with problem dimensionality.

As a further alternative, we have demonstrated in Chapter 3 [65] that gradient-only optimization is able to robustly and efficiently optimize the discontinuous objective functions that occur in shape optimization. As pointed out in Chapters 3 and 4, gradient-only optimization algorithms are conventional gradient based optimization algorithms - which invariably exploit not only first-order gradient information, but also zeroth order function value information - modified to no longer use function value information. Hence, the computational efficiency of gradient-only optimization algorithms is sometimes comparable to "conventional" gradient based optimization algorithms, provided that computationally efficient sensitivities are available.

In this chapter, we aim to extend the remeshing shape optimization strategy proposed in Chapter 2, by adding error indicators, with the objective of improving the accuracy of the computed structural response for a fixed number of degrees of freedom. In addition,

[1] Mesh topology is understood to refer to the number of nodes and nodal connectivity of a mesh.

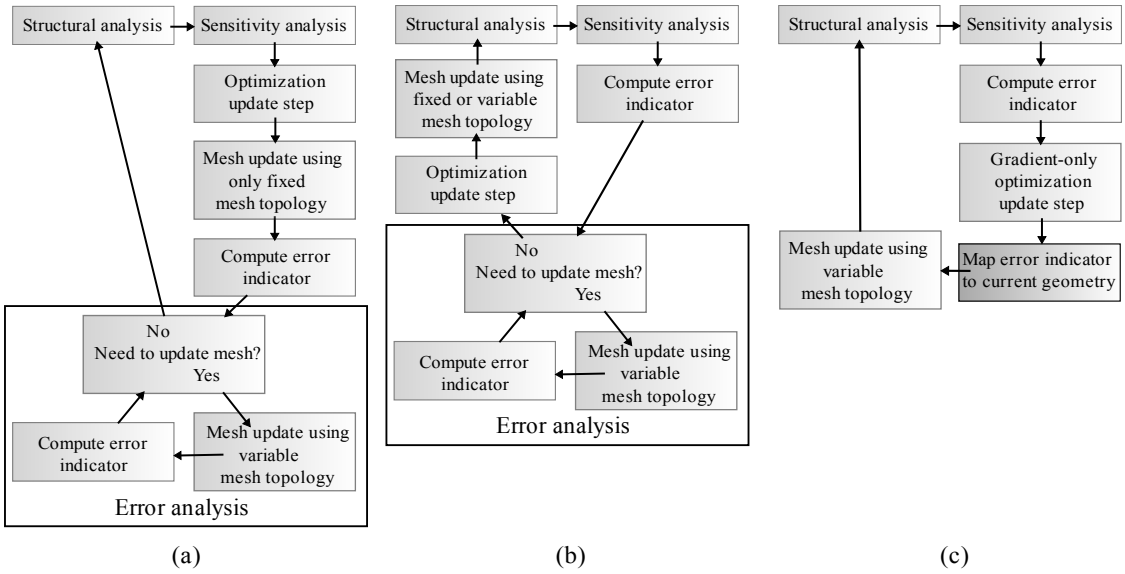


Figure 5.1: FE-error indicator integration into optimization

(semi) analytical sensitivities are made available for use in gradient-only optimization approaches. We can indeed incorporate error indicators freely, since we are able to robustly and efficiently optimize the discontinuous objective functions resulting from changes in the mesh topology.

Shape optimization may be a natural companion to *a posteriori* adaptive finite element mesh refinement, since both techniques share the computational burden of multiple analyses [16, 23, 30, 48]. A large portion of the computational burden associated with *a posteriori* adaptive finite element mesh refinement in once-off analyses is already accommodated for during shape optimization. *A posteriori* error indicators can be incorporated into finite element based shape optimization environments using two distinctly different approaches [48], namely whether changes in mesh *topologies* are allowed between optimization iterations or not.

Approaches that require the mesh topology to remain fixed between optimization iterations only update the mesh topology after the optimization update, as depicted in Figure 5.1(a). Hence, a single FEA is required for each candidate shape design if the error remains sufficiently small after a shape design update. Otherwise, multiple analyses are required per candidate shape design in order to reduce the error. Since the function values computed with different mesh topologies differ for the same candidate shape design, the optimization update may have to be repeated with the updated mesh topology [48]. This strategy may be efficient when design changes are small and no error control is required between design updates.

Alternatively, the mesh topology may be updated to control the discretization error during each optimization update, as depicted in Figure 5.1(b). Multiple finite element analyses may be required for each candidate shape design to control the discretization

error, i.e. to limit the size of the jump discontinuities resulting from different mesh topologies, allowing for the efficient use of classical gradient based algorithms [48].

In this chapter, we implement the latter approach, but relax the strict error control required when using classical gradient based algorithms, as shown in Figure 5.1(c). We are able to relax the error analysis and strict discretization error control for each candidate shape design, since gradient-only optimization allows us to robustly and efficiently optimize the resulting discontinuous cost functions, as we have previously demonstrated [65]. Instead of obtaining a converged error for each candidate shape design, we allow the error to converge as the shape designs converge [61].

Our proposed strategy requires only a single FEA for each candidate shape design. This is achieved by mapping the computed error indicator of a given shape geometry to the geometry obtained after an iteration of our gradient-only optimization algorithm. The mapping of the error indicator field between two shape geometries is merely a relocation of the nodal positions of the error indicator mesh from one shape geometry to the next using radial basis functions, as opposed to a linearization of the error indicator field between two shape geometries [10]. The advantage of this mapping is that the required number of computations are fewer than that required to compute a linearized error indicator field, or the actual error indicator field which would require a full FEA.

5.2 Shape optimization problem

The problem under consideration is the equality constrained shape optimization problem, for which the Lagrangian is given by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathcal{F}\left(\Omega(\boldsymbol{x})\right) + \sum_{j=1}^{m} \lambda_j g_j(\boldsymbol{x}), \ \boldsymbol{x} \in X \subseteq \mathbb{R}^n \text{ and } \boldsymbol{\lambda} \in \mathbb{R}^m, \tag{5.1}$$

where the objective function $\mathcal{F}\left(\Omega(\boldsymbol{x})\right)$ is a scalar function that depends on the geometry Ω of the structure, which in turn depends on the control variables $\boldsymbol{x}$ that describe the geometrical boundary $\partial\Omega$. The equality constraints $g_j(\boldsymbol{x}) = 0, \ j = 1, 2, \cdots, m$ are scalar functions of the control variables $\boldsymbol{x}$. For the sake of brevity, the cost function and the constraints will respectively be denoted by $\mathcal{F}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$; this notation will however imply dependency on $\Omega(\boldsymbol{x})$. We choose to represent the geometrical boundary $\partial\Omega$ by a simple piecewise linear interpolation between the control variables. However, Bezier curves or B-splines, etc. may of course also be used.

Normally, the saddle point of (5.1) is solved for using the dual formulation

$$\max_{\boldsymbol{\lambda}} \{ \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) \}. \tag{5.2}$$

We however solve (5.1) using the gradient-only dual formulation [65]

$$\max_{\boldsymbol{\lambda}}^{g}\{\min_{\boldsymbol{x}}^{g} L(\boldsymbol{x},\boldsymbol{\lambda})\}, \tag{5.3}$$

with $\max_{\boldsymbol{\lambda}}^{g}$ defined as follow: find $\boldsymbol{\lambda}$, such that

$$\nabla_{\boldsymbol{\lambda}}^{\mathrm{T}} L(\boldsymbol{x},\boldsymbol{\lambda}+\gamma_v \boldsymbol{v})\boldsymbol{v} \leq 0,\ \forall\ \boldsymbol{v} \in \mathbb{R}^m. \tag{5.4}$$

Similarly, $\min_{\boldsymbol{x}}^{g}$ is defined as follows: find $\boldsymbol{x}$, such that

$$\nabla_{\boldsymbol{x}}^{\mathrm{T}} L(\boldsymbol{x}+\delta_u \boldsymbol{u},\boldsymbol{\lambda})\boldsymbol{u} \geq 0, \forall\ \boldsymbol{u} \in \mathbb{R}^n \text{ such that } \boldsymbol{x}+\delta_u \boldsymbol{u} \in X, \tag{5.5}$$

with X the convex set of all possible solutions, $\nabla_{\boldsymbol{x}}$ the partial derivatives w.r.t. $\boldsymbol{x}$, $\nabla_{\boldsymbol{\lambda}}$ the partial derivatives w.r.t. $\boldsymbol{\lambda}$ and δ_u and γ_v real positive numbers. Note that we have only exploited gradient information of $L(\boldsymbol{x},\boldsymbol{\lambda})$.

5.3 Optimization algorithm

We will use the gradient-only sequential spherical approximation (SSA) algorithm presented by Wilke et. al. [65] to optimize the discontinuous shape optimization problem. For the sake of completeness and brevity, we merely outline the algorithm here (for details on the algorithm, and a motivation for using gradient-only optimization methods in the first place, the reader is referred to [65]):

1. **Initialization:** Select real constants $\epsilon > 0$, $\alpha > 1$, initial curvature $c^{\{0\}} > 0$ and initial point $[\boldsymbol{x}^{\{0\}}\ \ \boldsymbol{\lambda}^{\{0\}}]$. Set $k := 1$, $s := 0$.

2. **Gradient evaluation:** Compute $\nabla_{\boldsymbol{x}}^{\mathrm{T}} L([\boldsymbol{x}^{\{k\}}\ \ \boldsymbol{\lambda}^{\{k\}}])$.

3. **Approximate optimization:** Construct the local gradient-only approximate subproblem

$$\boldsymbol{\nabla}\tilde{f}^{\{k\}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}^{\mathrm{T}} L([\boldsymbol{x}^{\{k\}}\ \ \boldsymbol{\lambda}^{\{k\}}]) + \boldsymbol{H}^{\{k\}}(\boldsymbol{x}-\boldsymbol{x}^{\{k\}}) \tag{5.6}$$

   at $\boldsymbol{x}^{\{k\}}$, using $\boldsymbol{H}^{\{k\}} = c^{\{k\}}\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and

$$c^{\{k\}} = \frac{(\boldsymbol{x}^{\{k-1\}}-\boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\nabla_{\boldsymbol{x}}^{\mathrm{T}} L([\boldsymbol{x}^{\{k-1\}}\ \ \boldsymbol{\lambda}^{\{k\}}]) - \nabla_{\boldsymbol{x}}^{\mathrm{T}} L([\boldsymbol{x}^{\{k\}}\ \ \boldsymbol{\lambda}^{\{k\}}]))}{(\boldsymbol{x}^{\{k-1\}}-\boldsymbol{x}^{\{k\}})^{\mathrm{T}}(\boldsymbol{x}^{\{k-1\}}-\boldsymbol{x}^{\{k\}})}. \tag{5.7}$$

   (In an inner loop, use $c^{\{k\}}$ as calculated in Step 6(b)). Solve this subproblem analytically, to arrive at $\boldsymbol{x}^{\{k*\}}$.

4. **Evaluation:** Compute $\nabla_{\boldsymbol{x}}^{\mathrm{T}} L([\boldsymbol{x}^{\{k*\}}\ \ \boldsymbol{\lambda}^{\{k\}}])$.

5. **Test if $\boldsymbol{x}^{\{k*\}}$ is acceptable:** if

$$\nabla_{\boldsymbol{x}}^{\mathrm{T}} L([\boldsymbol{x}^{\{k*\}} \ \boldsymbol{\lambda}^{\{k\}}])(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}) \leq \boldsymbol{\nabla}^{\mathrm{T}} \tilde{f}(\boldsymbol{x}^{\{k*\}})(\boldsymbol{x}^{\{k*\}} - \boldsymbol{x}^{\{k\}}) = 0 \tag{5.8}$$

goto Step 7.

6. **Initiate an inner loop to effect conservatism:**

   (a) Set $s := s + 1$.

   (b) Set $c^{\{k\}} := \alpha c^{\{k\}}$.

   (c) Goto Step 3.

7. **Move to the new iterate:** Set $\boldsymbol{x}^{\{k+1\}} := \boldsymbol{x}^{\{k*\}}$.

8. **Update multiplier:** Set $\boldsymbol{\lambda}^{\{k+1\}} := \boldsymbol{\lambda}^{\{k\}} + \boldsymbol{\lambda_s}^{\{k+1\}}$, with $\boldsymbol{\lambda_s}^{\{k+1\}}$ the multiplier update step.

9. **Convergence test:** if $\|[\Delta \boldsymbol{x}^{\{k\}} \ \Delta \lambda^{\{k\}}]\| < \epsilon$, OR $\|\Delta \boldsymbol{x}^{\{i\}}\| < \epsilon, \ \forall \ i = \{k-4, k-3, \ldots, k\}$, OR $k = k_{max}$, stop[2].

10. **Initiate an additional outer loop:** Set $k := k + 1$ and goto Step 2.

5.4 Structural analysis

In shape optimization, the cost function $\mathcal{F}(\boldsymbol{x}) = \mathcal{F}\big(\boldsymbol{u}(\boldsymbol{\mathcal{X}}(\boldsymbol{x}))\big)$ is an explicit function of the nodal displacements $\boldsymbol{u}$, which in turn is a function of the discretized geometrical domain $\boldsymbol{\mathcal{X}}$. The discretized geometrical domain $\boldsymbol{\mathcal{X}}$ is described by the control variables $\boldsymbol{x}$ which represent the geometrical boundary $\partial\Omega$. The nodal displacements $\boldsymbol{u}$ are obtained by solving the approximate finite element equilibrium equations for linear elasticity

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{5.9}$$

where $\boldsymbol{K}$ represents the assembled structural stiffness matrix and $\boldsymbol{f}$ the consistent structural nodal loads, from which the unknown displacements $\boldsymbol{u}$ can be computed. From $\boldsymbol{u}$, we can then locally compute elemental stress fields

$$\hat{\boldsymbol{\sigma}}_e = \boldsymbol{C}\boldsymbol{B}_e\boldsymbol{u}_e, \tag{5.10}$$

with constitutive relationship $\boldsymbol{C}$, element kinematic relation $\boldsymbol{B}_e$ and element displacement $\boldsymbol{u}_e$. By combining the local stress fields $\hat{\boldsymbol{\sigma}}_e$ of adjacent elements, we obtain a global

[2]The notation used is $\Delta\phi^{\{k\}} = \phi^{\{k+1\}} - \phi^{\{k\}}$.

discontinuous stress field $\hat{\boldsymbol{\sigma}}$ over the entire structure, since inter-elemental stress continuity is not enforced. As the true stress field is continuous, error indicators may be recovered from the discontinuous stress field [70].

As said, shape optimization and *a posteriori* adaptive finite element refinement may naturally compliment each other, since both imply multiple FEAs. Instead of only conducting a FEA for each candidate shape design, we also recover error indicators from these FEAs. The recovered error from a given shape design is then used to discretize an updated shape design, causing the refinement strategy to converge as the shape design converges.

5.4.1 Recovery-based global error indicator

Although many recovery-based error indicators exist which range from so-called global to local indicators [69], we will herein opt for only the well-known Zienkiewicz-Zhu (ZZ) global error indicator [71]. We do so merely to avoid distraction - other indicators may equally well be used. The ZZ error indicator approximates the exact error in the energy norm by considering the difference in the energy norm of the smooth stress field $\breve{\boldsymbol{\sigma}}$ and the discrete stress field $\hat{\boldsymbol{\sigma}}$ as follows:

$$\|\boldsymbol{e}\|^2 = \int_\Omega (\breve{\boldsymbol{\sigma}} - \hat{\boldsymbol{\sigma}})^{\mathrm{T}} \boldsymbol{C}^{-1} (\breve{\boldsymbol{\sigma}} - \hat{\boldsymbol{\sigma}}) d\Omega. \tag{5.11}$$

The smooth stress field $\breve{\boldsymbol{\sigma}}$ is obtained from a least squares fit through the discrete nodal stress values. This requires a system of the size of the number of nodes to be solved. For the i^{th} element the square of the energy norm of the finite element solution $\|\hat{\boldsymbol{v}}_i\|^2$ is given by

$$\|\hat{\boldsymbol{v}}_i\|^2 = \boldsymbol{u}_i^T \boldsymbol{K}_i \boldsymbol{u}_i. \tag{5.12}$$

The corrected energy norm $||\boldsymbol{v}||$, which is used to approximate the exact energy norm, is given by

$$\|\boldsymbol{v}\|^2 = \|\hat{\boldsymbol{v}}\|^2 + \|\boldsymbol{e}\|^2, \tag{5.13}$$

where $\|\boldsymbol{e}\|^2$ and $\|\hat{\boldsymbol{v}}\|^2$ are computed by summing the elemental contributions, given by $\sum_i^r \|\boldsymbol{e}_i\|^2$ and $\sum_i^r \|\hat{\boldsymbol{v}}_i\|^2$, where r is the number of elements. Using the corrected energy norm $\|\boldsymbol{v}\|$, the average element error $\bar{e}$ is computed by taking a fraction of the root mean square of the corrected energy norm, defined as

$$\bar{e} = \iota \frac{\|\boldsymbol{v}\|}{\sqrt{r}}, \tag{5.14}$$

where ι represents the relative error tolerance. We choose to keep the number of nodes constant in our remeshing strategy, and as a result, we select $\iota = 1$. For our numerical

studies, we compute the global error η as

$$\eta = \sqrt{\frac{\|\boldsymbol{e}\|^2}{\|\boldsymbol{v}\|^2}}. \tag{5.15}$$

5.4.2 Refinement procedure

Using the computed error, we seek a refinement strategy to indicate spatial refinement (respectively de-refinement) of the mesh. For this, we modify the refinement procedure of Zienkiewicz and Zhu [71] to suit our r-refinement strategy: for the i^{th} element at the k^{th} iteration, we compute the refinement ratio $\xi_i^{\{k\}}$ as

$$\xi_i^{\{k\}} = \frac{\|\boldsymbol{e}_i^{\{k\}}\|}{\bar{e}^{\{k\}}}. \tag{5.16}$$

In turn, we use the refinement ratio $\xi_i^{\{k\}}$ to compute the ideal element length[3]

$$\hat{h}_i^{\{k\}} = \frac{h_i^{\{k-1\}}}{\left(\xi_i^{\{k\}}\right)^{1/p}}, \tag{5.17}$$

with $h^{\{0\}}$ chosen as the ideal element length of a uniform mesh for the first iteration. This also defines the initial number of nodes for our r-refinement strategy. Here, p is usually selected as the polynomial order of the shape functions away from singularities, and adjusted near singularities [71]. Our mesh generator naturally generates linear strain triangle (LST) elements, for which we found experimentally that $p < 4$ may result in oscillatory behavior. Hence, we have somewhat arbitrarily selected $p = 5$ herein [61]. We then smooth the discrete elemental scalar field $\hat{h}^{\{k\}}$ using nodal averaging to obtain a piece-wise continuous ideal element field $\tilde{h}^{\{k\}}$ described by a finite element interpolation. Finally, we normalize the continuous ideal element field $\tilde{h}^{\{k\}}$ to obtain

$$\breve{h}^{\{k\}} = \frac{\tilde{h}^{\{k\}}}{\bar{h}^{\{k\}}} \kappa h^{\{0\}}, \tag{5.18}$$

with $\bar{h}^{\{k\}}$ the average continuous ideal element length $\tilde{h}^{\{k\}}$ and κ a scaling factor. We select κ as constant, but allowing κ to vary (as some function of the initial area, current area, number of initial boundary nodes and current boundary) may well be beneficial. Finally, we limit the minimum ideal element length

$$\breve{h}^{\{k\}} = h_{\min} \; \forall \; \breve{h}^{\{k\}} < h_{\min}. \tag{5.19}$$

[3]Ideal element length refers to the "unloaded" truss lengths of the truss members in our truss structure analogous mesh generator [66] - also see Section 5.5.

5.5 Adaptive mesh generator

Our mesh generator solves for the equilibrium of a truss structure [42], which doubles as the finite element mesh, using the ideal element length field $h^{\{k\}}, k = 1, 2, 3, \ldots$ as the unloaded truss lengths, as opposed to directly optimizing the mesh according to some optimality criterion [37, 60]. It has been demonstrated [42, 65, 66] that this approach generates "good" meshes.

We start with an initial uniform mesh $\boldsymbol{\mathcal{X}}^{\{0\}}$ at $k = 0$, using a uniform ideal element length field $h^{\{0\}}$ [66]. After each analysis we compute the ideal element length field $\breve{h}^{\{k\}}$ using the refinement strategy described in Section 5.4.2. Recall that we then merely relocate the nodal positions of the computed ideal element length field to the new candidate shape design obtained from the optimization step, to avoid multiple analyses per candidate shape design, as illustrated in Figure 5.1(c). (We will describe the details of the mapping of the error field in Section 5.5.2.)

As with our previous mesh generator [66], we partition the mesh $\boldsymbol{\mathcal{X}}^{\{k\}}$ along the interior nodes $\boldsymbol{\mathcal{X}}^{\{k\}\Omega}$ and boundary nodes $\boldsymbol{\mathcal{X}}^{\{k\}\partial\Omega}$, which allows independent treatment of the boundary nodes $\partial\Omega^{\{k\}}$ and interior nodes $\Omega^{\{k\}}$. Superscript k denotes the iteration counter, which we will omit for the sake of brevity, unless we explicitly want to highlight the dependency on k.

The boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ are seeded according to the ideal element length field $\breve{h}^{\{k\}}$ along the geometrical boundary $\partial\Omega$, with nodes explicitly placed on the control variable locations $\boldsymbol{x}$. This ensures accurate representation of the defined geometrical domain Ω. Therefore $\boldsymbol{x} \subset \boldsymbol{\mathcal{X}}^{\partial\Omega}$, with $\partial\Omega$ described by a piece-wise linear interpolation of $\boldsymbol{x}$. The boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ remain fixed during the current iteration of the mesh generation process. We therefore *only* solve for $\boldsymbol{\mathcal{X}}^{\Omega}$ in finding the equilibrium of the truss structure

$$\boldsymbol{F}_{\Omega}\left(\mathcal{F}(\boldsymbol{\mathcal{X}}^{\Omega})\right) = \boldsymbol{0}. \tag{5.20}$$

The equilibrium of the truss structure is related to the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ via the force function

$$\mathcal{F}(\boldsymbol{\mathcal{X}}^{\Omega}) = \mathcal{F}\left(\boldsymbol{l}(\boldsymbol{\mathcal{X}}^{\Omega}), \boldsymbol{l_0}(\breve{h}^{\{k\}}, \boldsymbol{\mathcal{X}}^{\Omega})\right) = \mathcal{K}(\boldsymbol{l_0} - \boldsymbol{l}), \tag{5.21}$$

which depends on the constant spring stiffness $\mathcal{K}$, the length of the truss members $\boldsymbol{l}(\boldsymbol{\mathcal{X}}^{\Omega})$ and the undeformed truss lengths $\boldsymbol{l_0}\left(\breve{h}^{\{k\}}, \boldsymbol{\mathcal{X}}^{\Omega}\right)$. The undeformed truss lengths $\boldsymbol{l_0}\left(\breve{h}^{\{k\}}, \boldsymbol{\mathcal{X}}^{\Omega}\right)$ depend on the ideal element length field $\breve{h}^{\{k\}}$, as well as the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$, since the ideal element length field $\breve{h}^{\{k\}}$ is evaluated at the midpoint of each truss member.

However, the ideal element length field $\breve{h}^{\{k\}}(\boldsymbol{e})$ is taken as a constant background field [10, 27, 50, 61] during the mesh generation process, i.e. we do not recompute the error field or linearize the error field when the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ of the mesh vary. Consequently, the

dependency of the ideal element length field $\breve{h}^{\{k\}}(\boldsymbol{e})$ on the spatially varying *a posteriori* error field $\boldsymbol{e}$ is constant and the sensitivity zero. Lastly, the displacement field $\boldsymbol{u}(\boldsymbol{\mathcal{X}}^{\Omega})$ depends on the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$.

The reduced truss system in Eq. (5.20) is solved directly via the quadratically convergent Newton's method, which is given by

$$\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} \Delta \boldsymbol{\mathcal{X}}^{\Omega} = -\boldsymbol{F}_{\Omega}. \tag{5.22}$$

The update of nodal coordinates is given by

$$\boldsymbol{\mathcal{X}}^{\Omega}_{n+1} = \boldsymbol{\mathcal{X}}^{\Omega}_{n} + \Delta \boldsymbol{\mathcal{X}}^{\Omega}, \tag{5.23}$$

and for a constant ideal element length background field the consistent tangent $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}}$ is given by

$$\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} = \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{l}} \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} + \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{l_0}^{\{k\}}} \frac{\partial \boldsymbol{l_0}^{\{k\}}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}}. \tag{5.24}$$

We obtain quadratic convergence using Newton's method.

5.5.1 Boundary nodes

A first approach to accommodate a spatially varying ideal element length field may be to merely change the ideal element length and boundary spacing, while the number of boundary nodes and interior nodes follow from our previous uniform remeshing strategy [66]. However, limited improvements are achieved using this naive strategy.

In this study we let the number of boundary nodes follow from the spatially varying ideal element length field by first placing nodes along the boundary $\boldsymbol{\mathcal{X}}^{\partial\Omega}$. Since we aim to keep the number of nodes constant, the remaining nodes are seeded in the interior domain $\boldsymbol{\mathcal{X}}^{\Omega}$. Although this strategy may seem simplistic, we found other strategies to determine the number of boundary nodes, like using ratios of the circumference to interior area together with the error along the circumference to the interior, susceptible to oscillations.

As stated before the boundary nodes remain fixed once they have been placed along the boundary, which reduces the size of the Newton system when solving for the truss equilibrium. Consequently, an increase in the number of boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ results in a decrease in the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ and vice versa, as we want to keep the total number of nodes fixed. To reduce the computational effort, we use the nodes that describe the ideal element length field $\breve{h}^{\{k\}}$ as an initial guess for our interior nodes. To keep the total number of nodes constant, we need to either remove or add nodes to those nodes used in representing $\breve{h}^{\{k\}}$. We do so by ranking the nodes and elements according to their error densities. We remove nodes by starting with nodes with smaller error densities, and add nodes by introducing nodes at the centroids of elements with higher element error

densities.

5.5.2 Mapping the error field between candidate shape designs

The ideal element length field $\breve{h}^{\{k\}}$ for the k^{th} iteration is obtained by mapping the computed ideal element length field after the analysis of the $(k-1)^{\text{th}}$ iteration to the candidate shape design for the k^{th} iteration. We achieve this by merely mapping the nodal positions, without requiring connectivity information, using radial basis functions (RBF). We therefore have total flexibility on whether we want to use or discard the nodal connectivity of the previous geometry. In this study we pay the computational penalty of re-triangulation at every iteration to keep our strategy unsophisticated, while allowing for large shape changes.

The details of the mapping strategy using radial basis functions are outlined in the Appendix.

5.6 Sensitivity analysis

Recall that the cost function $\mathcal{F}(\boldsymbol{u}(\boldsymbol{\mathcal{X}}(\boldsymbol{x})))$ is an explicit function of the nodal displacements. Specifically, in all the examples herein, the cost function $\mathcal{F}$ is the nodal displacement at the point where a point load F is applied, expressed as

$$\mathcal{F}(\boldsymbol{u}(\boldsymbol{x})) = \boldsymbol{u}_{\text{F}}(\boldsymbol{x}). \tag{5.25}$$

The displacement field $\boldsymbol{u}(\boldsymbol{x})$ depends on the discretized geometrical domain $\boldsymbol{\mathcal{X}}$, which is obtained by solving for the nodal positions of a truss structure at equilibrium. The ideal element lengths $\breve{h}^{\{k\}}$ of the truss structure are obtained from the error analysis discussed in Section 5.4. Then, the sensitivity of the displacement $\boldsymbol{u}_{\text{F}}$ w.r.t. the control variables $\boldsymbol{x}$ is obtained by computing

$$\frac{d\boldsymbol{u}_{\text{F}}}{d\boldsymbol{x}} = \frac{d\boldsymbol{u}_{\text{F}}}{d\boldsymbol{\mathcal{X}}}\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}. \tag{5.26}$$

The computation of $\frac{d\boldsymbol{u}_{\text{F}}}{d\boldsymbol{\mathcal{X}}}$ is obtained by direct differentiation of the finite element equilibrium equations $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$, given by

$$\frac{d\boldsymbol{K}}{d\boldsymbol{\mathcal{X}}}\boldsymbol{u} + \boldsymbol{K}\frac{d\boldsymbol{u}}{d\boldsymbol{\mathcal{X}}} = \frac{d\boldsymbol{f}}{d\boldsymbol{\mathcal{X}}}. \tag{5.27}$$

For the fixed applied external loads $\boldsymbol{f}$ we will restrict ourselves to in this study, $\frac{d\boldsymbol{f}}{d\boldsymbol{\mathcal{X}}} = \boldsymbol{0}$ and (5.27) reduces to

$$\boldsymbol{K}\frac{d\boldsymbol{u}}{d\boldsymbol{\mathcal{X}}} = -\frac{d\boldsymbol{K}}{d\boldsymbol{\mathcal{X}}}\boldsymbol{u}. \tag{5.28}$$

We compute $\frac{d\boldsymbol{K}}{d\boldsymbol{\mathcal{X}}}$ by direct differentiation of the analytical stiffness matrices for the linear strain triangular elements [57, 66], which allows to solve for $\frac{d\boldsymbol{u}}{d\boldsymbol{\mathcal{X}}}$, to then obtain $\frac{d\boldsymbol{u}_{\mathrm{F}}}{d\boldsymbol{\mathcal{X}}}$.

The sensitivities of the nodal coordinates $\boldsymbol{\mathcal{X}}$ w.r.t. the control variables $\boldsymbol{x}$, $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{x}}$, are obtained by differentiating the truss structure equilibrium equations from the mesh generation in Section 5.5. Recall that we partitioned $\boldsymbol{\mathcal{X}}$ along the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ and boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$. Also recollect that the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$ are seeded according to the ideal element length field $\breve{h}^{\{k\}}$ along the geometrical boundary $\partial\Omega$, and that they remain fixed during the mesh generation process. Hence, the equilibrium of the truss structure $\boldsymbol{F}_{\Omega}$ only depends implicitly on the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$.

Consider the dependency of the equilibrium equations $\boldsymbol{F}_{\Omega}$ on $\boldsymbol{x}$:

$$\boldsymbol{F}_{\Omega}\left(\boldsymbol{l}(\boldsymbol{\mathcal{X}}^{\Omega}(\boldsymbol{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\breve{h}^{\{k\}}(\boldsymbol{x}))),\boldsymbol{l_0}(\breve{h}^{\{k\}}(\boldsymbol{x}),\boldsymbol{\mathcal{X}}^{\Omega}(\boldsymbol{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\boldsymbol{x}))\right)=\boldsymbol{0}. \tag{5.29}$$

The equilibrium equations $\boldsymbol{F}_{\Omega}$ depend on the deformed lengths $\boldsymbol{l}(\boldsymbol{\mathcal{X}}^{\Omega},\boldsymbol{\mathcal{X}}^{\partial\Omega})$ and undeformed lengths $\boldsymbol{l_0}(\breve{h}^{\{k\}}(\boldsymbol{x}),\boldsymbol{\mathcal{X}}^{\Omega}(\boldsymbol{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\boldsymbol{x}))$ of the truss members. The truss member lengths $\boldsymbol{l}(\boldsymbol{\mathcal{X}}^{\Omega},\boldsymbol{\mathcal{X}}^{\partial\Omega})$ in turn depend on the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ and boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$. The undeformed lengths $\boldsymbol{l_0}(\breve{h}^{\{k\}},\boldsymbol{\mathcal{X}}^{\partial\Omega})$ are evaluated at the midpoints of the truss members, which depend on the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ and the boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega}$. In addition, the ideal element length field $\breve{h}^{\{k\}}(\boldsymbol{x})$ changes as a function of $\boldsymbol{x}$ due to the RBF mapping. By taking the derivative of (5.29) w.r.t. to the control variables $\boldsymbol{x}$ we obtain

$$\begin{aligned}\frac{d\boldsymbol{F}_{\Omega}}{d\boldsymbol{x}}=&\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l}}\frac{\partial\boldsymbol{l}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\frac{\partial\boldsymbol{\mathcal{X}}^{\Omega}}{\partial\boldsymbol{x}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l}}\frac{\partial\boldsymbol{l}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}{\partial\breve{h}^{\{k\}}}\frac{\partial\breve{h}^{\{k\}}}{\partial\boldsymbol{x}}\\&+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\breve{h}^{\{k\}}}\frac{\partial\breve{h}^{\{k\}}}{\partial\boldsymbol{x}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\frac{\partial\boldsymbol{\mathcal{X}}^{\Omega}}{\partial\boldsymbol{x}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}{\partial\boldsymbol{x}}=\boldsymbol{0},\end{aligned} \tag{5.30}$$

to give

$$\begin{aligned}\left(\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l}}\frac{\partial\boldsymbol{l}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\right)\frac{\partial\boldsymbol{\mathcal{X}}^{\Omega}}{\partial\boldsymbol{x}}=&-\left(\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l}}\frac{\partial\boldsymbol{l}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}{\partial\breve{h}^{\{k\}}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\breve{h}^{\{k\}}}\right)\frac{\partial\breve{h}^{\{k\}}}{\partial\boldsymbol{x}}\\&-\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}{\partial\boldsymbol{x}}.\end{aligned} \tag{5.31}$$

From (5.31) we may solve for $\frac{\partial\boldsymbol{\mathcal{X}}^{\Omega}}{\partial\boldsymbol{x}}$, when $\left(\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l}}\frac{\partial\boldsymbol{l}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\right)$ and the right-hand side of (5.31) are known. We compute the right-hand side with a finite difference perturbation, and recall that $\left(\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l}}\frac{\partial\boldsymbol{l}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{l_0}}\frac{\partial\boldsymbol{l_0}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\right)$ is available from the Newton update, as noted in Section 5.5. Once we have solved for $\frac{\partial\boldsymbol{\mathcal{X}}^{\Omega}}{\partial\boldsymbol{x}}$, we obtain $\frac{\partial\boldsymbol{\mathcal{X}}}{\partial\boldsymbol{x}}$ as the union between $\frac{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}{\partial\boldsymbol{x}}$ and $\frac{\partial\boldsymbol{\mathcal{X}}^{\Omega}}{\partial\boldsymbol{x}}$, where $\frac{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}{\partial\boldsymbol{x}}$ is obtained numerically.

5.7 Numerical study

In our numerical studies, we will consider two remeshing strategies. Firstly, we use a remeshing strategy which we will refer to as *uniform*, which uses an ideal element length field that is spatially uniform, and also kept constant as the optimization iterations progress [66]. Secondly, we will use the newly developed remeshing strategy presented herein, which is characterized by a spatially varying ideal element field that changes as the optimization iterations progress, to which we will refer as *adaptive*.

The parameters used in the remeshing strategies, the finite element analyses and the optimization algorithm are as follows. For the adaptive remeshing strategy we set the mesh refinement parameter p in (5.17) to 5, while the minimum element length $h_{\min}$ is selected as $0.1h^{\{0\}}$, unless otherwise stated. The material considered is an orthotropic Boron-Epoxy in a tape lay-out, i.e. the fibers are all aligned in a single direction, aligned with the global x-axis. In other words, we do not determine an optimal fiber orientation. We assume plane stress conditions and use classical laminate theory (CLT). The material properties used are a longitudinal Young's modulus of $\mathrm{E}_1 = 228$ GPa, a transverse Young's modulus of $\mathrm{E}_2 = 145$ GPa, and a shear modulus of $G_{12} = 48$ GPa. The last independent parameter in CLT is Poisson ratio $\nu_{12} = 0.23$, since ν_{21} follows from the symmetry relation $\mathrm{E}_1\nu_{21} = \mathrm{E}_2\nu_{12}$.

The selected parameters for the gradient-only conservative sequential spherical approximation algorithm [65] are the curvature factor $\alpha = 2$, initial curvature $c^{\{0\}} = 1$, convergence tolerance $\epsilon = 10^{-4}$, and a maximum number of outer iterations $k_{\max} = 300$. The Lagrange multiplier update step is selected as $\boldsymbol{\lambda_s}^{\{k+1\}} = \nabla^{\mathrm{T}}_{\boldsymbol{\lambda}} L([\boldsymbol{x}^{\{k+1\}} \quad \boldsymbol{\lambda}^{\{k\}}])$. We also limit the maximum step size to 1, which was experimentally found to result in good convergence rates, but this value will in general of course strongly depend on scaling of the problem. Before we proceed, we first validate the (semi) analytical sensitivities and study the convergence rates of the meshing strategies.

5.7.1 Gradient sensitivity comparison

We compare our analytical sensitivities to numerical sensitivities obtained with the forward finite difference method. We compute the sensitivity of the displacement at the point of load application (u_{F}) w.r.t. the indicated control variables of the bow-tie structure [66] depicted in Figure 5.2. The control variables are linearly spaced along the top and bottom with the relevant dimensions as indicated in Figure 5.2.

Calculation of the finite difference values is done without Delaunay triangulation steps, to avoid the introduction of discontinuities (due to the addition or removal of nodes) in the finite difference sensitivity analysis. In Table 5.1, we tabulate the sensitivities w.r.t. the control variables for a spatially varying ideal element length field, which confirms that our computations are correct and accurate. The spatially varying ideal element length

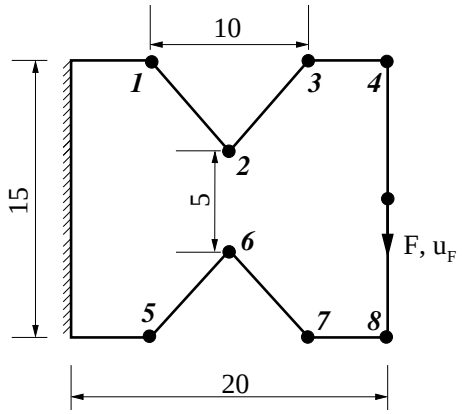


Figure 5.2: Bow-tie structure used to validate the (semi) analytical sensitivities and study the convergence behavior of the remeshing strategies.

Table 5.1: Analytical and forward finite difference sensitivities calculated for the bow-tie structure depicted in Figure 5.2.

Point	Analytical ($\times 10^{-3}$)	Numerical ($\times 10^{-3}$)	Point	Analytical ($\times 10^{-3}$)	Numerical ($\times 10^{-3}$)
1	-0.043653	-0.043653	5	0.044330	0.044330
2	-1.044313	-1.044312	6	1.073200	1.073201
3	-0.045178	-0.045178	7	0.028182	0.028182
4	-0.211877	-0.211875	8	0.136393	0.136392

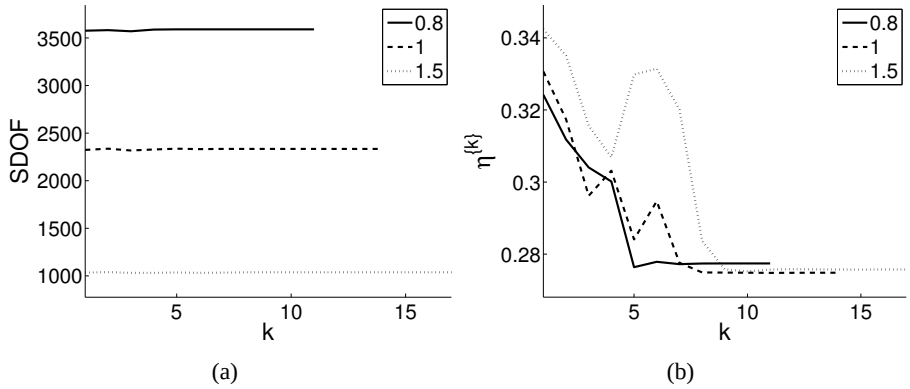


Figure 5.3: (a) System degrees of freedom (SDOF) and (b) global error $\eta^{\{k\}}$ for the mesh convergence study on the bow-tie structure for initial uniform element lengths $h_0 = \{1.5, 1, 0.8\}$.

field was obtained after 10 remeshing update iterations using an initial undeformed truss length of $h_0 = 2$. The numerical gradients are computed with a perturbation of 10^{-6}.

5.7.2 Convergence rates

Again consider the bow-tie structure depicted in Figure 5.2. The structure is meshed using three initial undeformed truss lengths $h_0 = \{1.5, 1, 0.8\}$. The convergence criterion for the error indicator is given by

$$\frac{\eta^{\{k\}} - \eta^{\{k-1\}}}{\eta^{\{k\}}} < 10^{-6}. \tag{5.32}$$

Results are presented in Figure 5.3, with the system degrees of freedom (SDOF) depicted in Figure 5.3(a) and the global error indicator depicted in Figure 5.3(b). From Figure 5.3(a) it is clear that the system degrees of freedom remain practically constant as the iterations progress. Figure 5.3(b) reveals that the final global error $\eta^{\{\max(k)\}}$ is lower than the global error of the initial uniform mesh $\eta^{\{1\}}$, for each of the three initial undeformed truss length choices. We also observe that the required number of refinement iterations reduces as the system degrees of freedom increase. Note that the global errors $\eta^{\{k\}}$ for the various undeformed truss lengths cannot be compared with each other, since each uses a different $\breve{\boldsymbol{\sigma}}$ field. For each of the initial uniform element lengths $h_0 = \{1.5, 1, 0.8\}$, we depict the initial meshes in Figure 5.4(a)-(c), the final meshes in Figure 5.4(d)-(f) and the ideal element length fields in Figure 5.4(g)-(i).

We depict the convergence of the displacements of the bow-tie structure in Figure 5.5. To do so, we approximate the analytical solution u_{F}^{*} using Richardson's extrapolation method [29], since an analytical solution is not available for this problem. For Richardson's extrapolation method we have used the initial uniform element lengths $h_0 = \{1.6, 0.8, 0.4\}$ to compute uniform meshes, with h_0 representative of the average

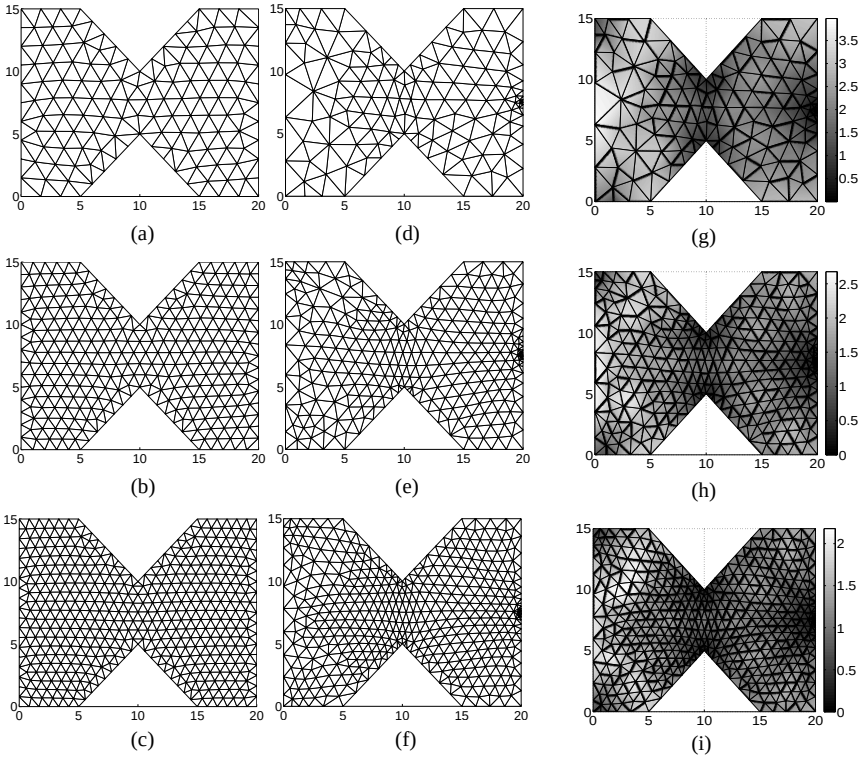


Figure 5.4: Convergence study showing (a)-(c) the initial mesh, (d)-(f) the final mesh and (g)-(i) the final ideal element length field of the bow-tie structure for various initial uniform element lengths $h_0 = \{1.5, 1, 0.8\}$.

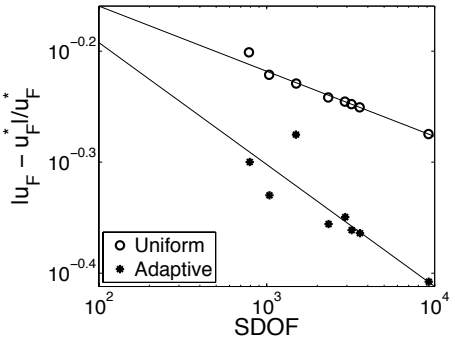


Figure 5.5: Approximated displacement convergence rate for the bow-tie structure problem using the uniform and adaptive mesh generators.

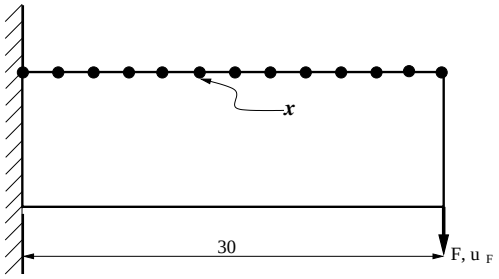


Figure 5.6: Initial geometry of the cantilever beam using 13 control points $\boldsymbol{x}$.

element length. To approximate the asymptotic convergence rate, we fit a straight line in a least squares sense through the four data points with the highest system degrees of freedom. As shown, the error of the adaptive mesh is less than that of the uniform mesh for a given number of degrees of freedom, while the convergence rate is superior.

5.7.3 Cantilever beam

Next, we progress to the equality constrained design of the orthotropic cantilever beam depicted in Figure 5.6. The structure has a predefined length of 30 mm and a thickness of 1 mm. A point load F of 10 N acts at the bottom right corner of the structure. The boundary of the structure is controlled by the 13 control points or design variables $\boldsymbol{x}$ that can only move vertically. The boundary is linearly interpolated between the control points and the control points are linearly spaced along the top of the cantilever beam. We minimize the displacement u_{F} at the point of load application, subject to an equality constraint on volume, expressed as $V(\boldsymbol{x}) = V_0$, with $V_0 = 150\ \mathrm{mm}^3$, the prescribed volume of the structure.

Convergence histories for the value of the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, the constraint function $|g(\boldsymbol{x}^{\{k\}})|$, the Lagrange multiplier $\lambda^{\{k\}}$ and the system degrees of freedom are depicted in Figure 5.7(a)-(d) for the uniform and adaptive mesh generators. (We have used an initial ideal element length of 1.05 for the uniform mesh generator and 1 for the adaptive mesh generator to get a comparable number of system degrees of freedom for the converged shapes.)

The required number of iterations and final designs are comparable. The system degrees of freedom of the uniform remeshing strategy changes as the geometry varies, since the defined geometrical domain changes while the uniform mesh generator maintains a constant element length. The number of system degrees of freedom of our adaptive remeshing strategy is roughly constant; the small variations present being the result of nodes being eliminated during convergence of the mesh generator. The interesting aspects of this example are depicted in Figure 5.8. The initial and final designs are respectively depicted in Figure 5.8(a)-(b), and Figure 5.8(c)-(d), with the final ideal element length fields depicted in Figure 5.8(e)-(f), for the uniform and adaptive mesh

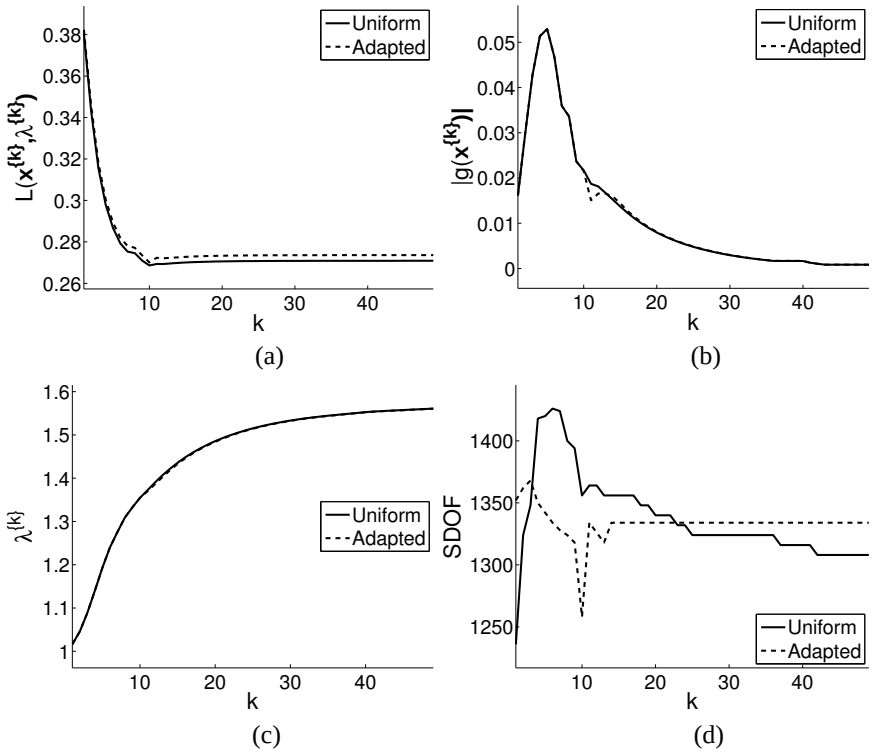


Figure 5.7: The cantilever beam convergence histories of (a) the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, (b) absolute value of the constraint function $g(\boldsymbol{x}^{\{k\}})$, (c) Lagrange multiplier $\lambda^{\{k\}}$, and (d) system degrees of freedom (SDOF) for a uniform and adapted mesh using initial ideal element lengths h_0 of respectively 1.05 and 1.

generators respectively - note the superiority of the latter mesh. The converged designs depicted in Figure 5.8(c)-(d) reflect the parabolic shape known as the analytical solution of the equivalent beam problem.

5.7.4 Michell structure

Next, we consider the equality constrained design of the orthotropic Michell-like structure depicted in Figure 5.9. The structure also has a predefined length of 30 mm and thickness of 1 mm, and a point load F of 10 N acts at the center bottom of the structure. The boundary of the structure is controlled by the 16 control points $\boldsymbol{x}$, which can only move vertically. Again the boundary is linearly interpolated between the control points and the control points are linearly spaced along the top and the bottom of the structure. We minimize the displacement u_{F} at the point of load application, subject to an equality constraint on volume, expressed as $V(\boldsymbol{x}) = V_0$, with $V_0 = 75$ mm^3, the prescribed volume of the structure.

Convergence histories for the value of the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, the constraint function $|g(\boldsymbol{x}^{\{k\}})|$, the Lagrange multiplier $\lambda^{\{k\}}$ and the system degrees of freedom are depicted in Figure 5.10(a)-(d) for the uniform and adaptive mesh generators. We have used an initial ideal element length of 0.7 for the uniform mesh generator and 0.8 for the

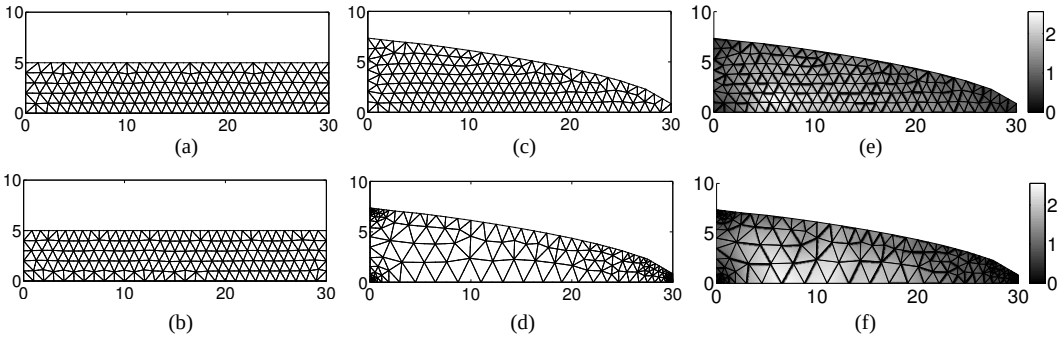


Figure 5.8: Initial (a)-(b) and final (c)-(d) designs of the cantilever beam with the associated final ideal element length field (e)-(f), for a uniform and adapted mesh using initial ideal element lengths h_0 of respectively 1.05 and 1.

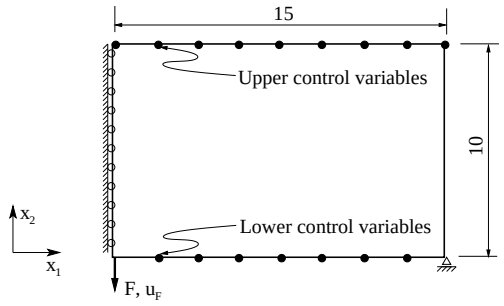


Figure 5.9: Initial geometry of half the Michell-like structure using 16 control points $\boldsymbol{x}$.

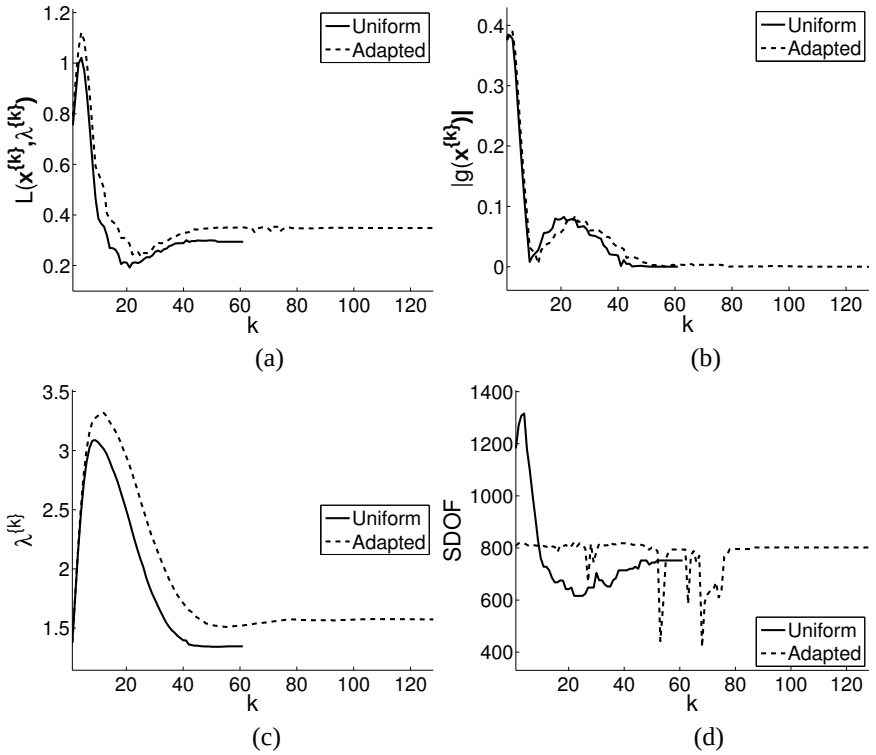


Figure 5.10: The Michell structure convergence histories of (a) the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, (b) absolute value of the constraint function $g(\boldsymbol{x}^{\{k\}})$, (c) Lagrange multiplier $\lambda^{\{k\}}$, and (d) system degrees of freedom (SDOF) for a uniform and adaptive mesh using initial ideal element lengths h_0 of respectively 0.7 and 0.8.

adaptive mesh generator.

The required number of iterations for the uniform mesh generator is roughly half that required for the adaptive mesh generator, but the latter mesh is superior. Again the system degrees of freedom of the uniform remeshing strategy changes as the geometry varies, while the initial and final system degrees of freedom of the adaptive remeshing strategy remains almost constant.

The initial and final designs are respectively depicted in Figure 5.11(a)-(b) and Figure 5.11(c)-(d), with the final ideal element length fields depicted in Figure 5.11(e)-(f), for the uniform and adaptive mesh generators. While different, the converged designs depicted in Figure 5.11(e)-(f) are similar and compare well with results obtained during previous studies [20, 66].

5.7.5 Spanner design

Finally, we consider the shape design of the full spanner problem presented in Figure 5.12, which is subjected to multiple load cases. The objective is to minimize $\frac{1}{2}(u_{\text{FA}} - u_{\text{FB}})$, with u_{FA} and u_{FB} the vertical displacements at the point of load application, for the two independent load cases F_A and F_B respectively. The spanner is subjected to an equality constraint on volume, expressed as $V(\boldsymbol{x}) = V_0$, with $V_0 = 70$ mm^3, the prescribed volume

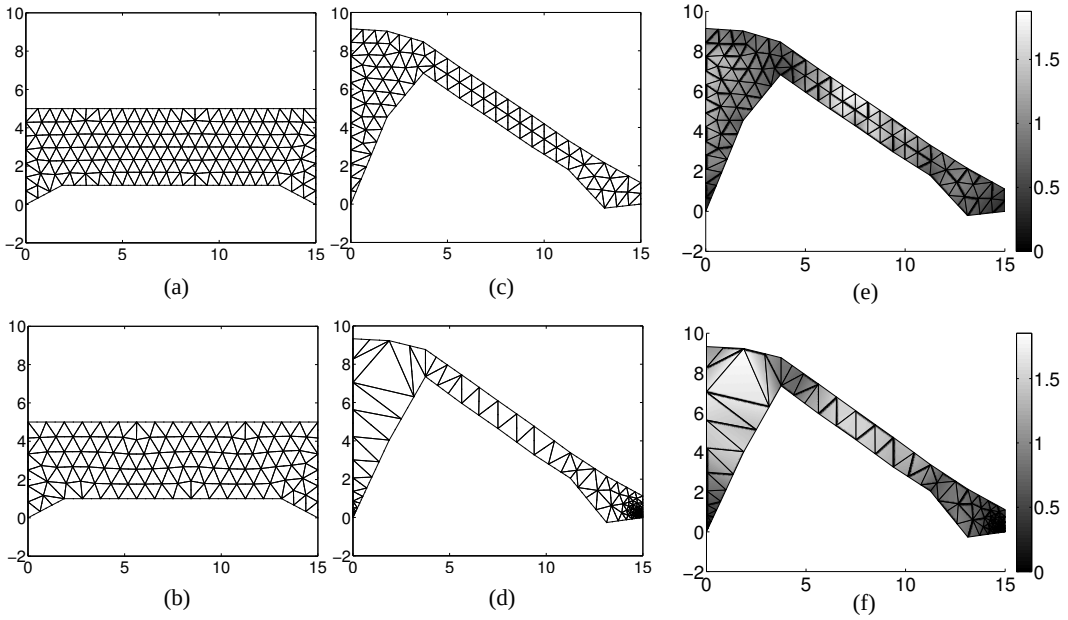


Figure 5.11: Initial (a)-(b) and final (c)-(d) designs of the Michell structure with the associated final ideal element length field (e)-(f), for a uniform and adapted mesh using initial ideal element lengths h_0 of respectively 0.7 and 0.8.

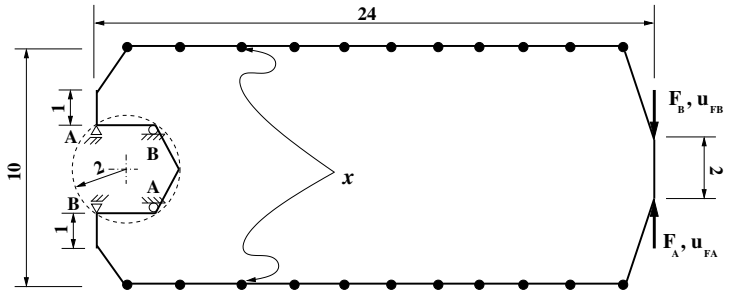


Figure 5.12: Initial geometry and loads of the full spanner problem using 22 control points $\boldsymbol{x}$.

of the structure.

The upper and lower boundaries of the geometry are described using 11 control points each. In addition, the control points are linearly spaced along the length of the spanner. The structure has a predefined length of 24 mm and thickness of 1 mm. The magnitude of the point loads F_A and F_B is 1 N each. Symmetry is *not* enforced; deviations from symmetry may be used to qualitatively evaluate the obtained designs since this problem should result in a symmetric geometry. The ideal element length field $\breve{h}^{\{k\}}$ for the mesh is obtained by nodal averaging of the ideal element length fields obtained from the different load cases.

Convergence histories for the value of the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, the constraint function $|g(\boldsymbol{x}^{\{k\}})|$, the Lagrange multiplier $\lambda^{\{k\}}$ and the system degrees of freedom are depicted in Figure 5.13(a)-(d) for the uniform and adaptive mesh generators. This time, the required number of iterations for the uniform mesh generator are slightly more than that required for the adaptive mesh generator. Again the system degrees of freedom of the uniform remeshing strategy changes as the geometry varies, while the system degrees of freedom of our adaptive remeshing strategy remains roughly constant after an initial unstable 80 iterations. The results depicted in Figure 5.14 compare well with results obtained in previous studies [25, 63]. Due to changes in the SDOF, as depicted in Figure 5.13(d), some oscillatory behavior is observed in the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$ within the first 70 iterations, see Figure 5.13(a).

5.8 Conclusions

In this study we successfully extended our uniform remeshing strategy [66] to incorporate the well known Zienkiewicz and Zhu global error indicator and refinement strategy. As demonstrated on a bow-tie structure we significantly improve on the quality of the results obtained with uniform meshes.

In addition, we showed how gradient-only optimization allows us to efficiently incorporate error indicators and refinement strategies, since we only require a single finite element analysis followed by *a posteriori* error computation for each candidate shape design, without sacrificing optimization robustness. We demonstrated our strategy on three equality constrained example problems.

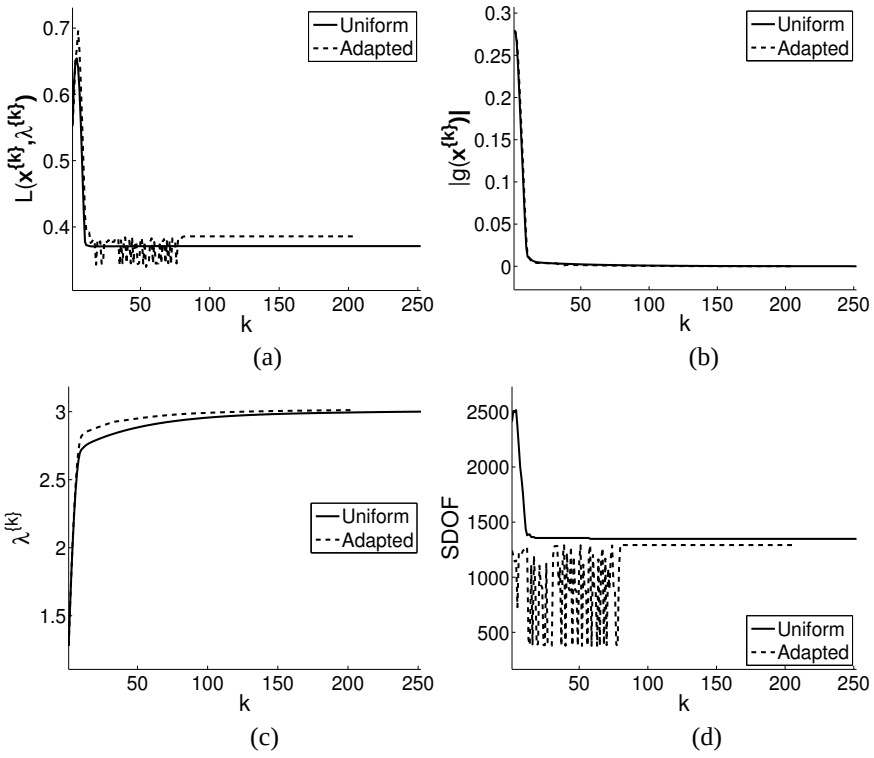


Figure 5.13: The full spanner convergence histories of (a) the Lagrangian $L(\boldsymbol{x}^{\{k\}}, \lambda^{\{k\}})$, (b) absolute value of the constraint function $g(\boldsymbol{x}^{\{k\}})$, (c) Lagrange multiplier $\lambda^{\{k\}}$, and (d) system degrees of freedom (SDOF) for a uniform and adaptive mesh using initial ideal element lengths h_0 of respectively 0.7 and 1.

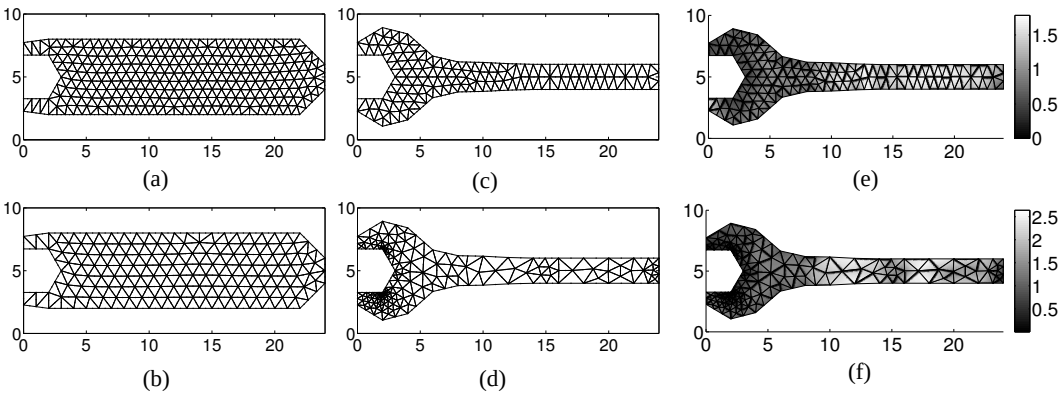


Figure 5.14: Initial (a)-(b) and final (c)-(d) designs of the full spanner with the associated final ideal element length field (e)-(f), for a uniform and adapted mesh using initial ideal element lengths h_0 of respectively 0.7 and 1.

CHAPTER 6

Conclusion and recommendations

This chapter summarises the findings of this study and makes recommendations for further research and investigation.

We have applied gradient based optimization techniques to shape design problems. In doing so, we have created a novel unstructured remeshing shape optimization environment, based on a truss structure analogy. The remeshing environment is quadratically convergent in solving for the equilibrium positions of the truss structure.

As may be expected, the objective function value in general decreases as the number of control points are increased. This is a direct result of the number of possible design configurations increasing. However, due to unstructured remeshing, non-physical step or jump discontinuities may be introduced into the optimization problem. These discontinuities arise when (partial) differential equations are discretized using non-constant methods: the functions become discontinuous but gradient information is computable everywhere since every point has an associated discretization for which (semi-) analytical sensitivities can be calculated. Although the magnitude of these discontinuities decreases with mesh refinement, their number increases. For the gradient based algorithms, the severity of the anomaly is alleviated as the mesh is refined. Polynomial refinement e.g. linear strain triangles, further decreases the magnitude of the discontinuities.

Although these local minima may be overcome efficiently and effectively using a simple multi-start strategy, the computational efficiency and robustness of multi-start strategies may be improved on using gradient-only optimization strategies.

To illustrate, we proposed gradient-only implementations of the BFGS algorithm and a SAO algorithm for discontinuous problems, and applied these algorithms to a selection of problems of practical interest, both unconstrained and constrained. These are the design of a heat exchanger fin, the shape design of an orthotropic Michell-like structure,

and a material identification study using a modified Voce law, all discretized using non-constant methods. In each instance, the gradient-only based algorithms found superior solutions to the classical methods that use both function and gradient information.

As opposed to surrogate methods based on design of experiment (DOE) techniques, which scale poorly, gradient-only algorithms based on classical optimization algorithms that scale well, may also be expected to scale well (provided the gradient computations scale well); this may well become an important application of gradient-only methods. Another envisaged application of gradient-only algorithms are any problem for which gradient computations are inexpensive.

In addition, we successfully extended the uniform remeshing strategy presented in Chapter 2 to incorporate the well known Zienkiewicz and Zhu global error indicator and refinement strategy. It significantly improves on the quality of the results obtained with uniform meshes. We showed how gradient-only optimization allows us to incorporate error indicators and refinement strategies efficiently, since we only require a single finite element analysis followed by *a posteriori* error computation for each candidate shape design, without sacrificing optimization robustness.

The implications of our approach are that variable discretization strategies, which are so important in numerical discretization methods, may be used in combination with efficient local optimization algorithms, notwithstanding the fact that these strategies themselves introduce step discontinuities.

Among others, future endeavors should in our opinion concentrate on the inclusion of constraint functions, in particular step discontinuous constraint functions e.g. stress constraints, as well as the reduction of the required computational effort.

An investigation of whether linearizing the error indicator field improves the convergence of the gradient-only optimization strategies should in our opinion also be conducted.

Bibliography

[1] Allaire, G, Jouve, F and Toader, A (February 2004), "Structural optimization using sensitivity analysis and a level-set method," *Journal of Computational Physics*, 194 (1), 363–393, ISSN 0021-9991.

[2] Avriel, M (2003), *Nonlinear programming Analysis and Methods*, Dover.

[3] Barthelemy, JF and Haftka, RT (1993), "Approximation concepts for optimum structural design - a review," *Struct. Opt.*, 5, 129–144.

[4] Bazaraa, MS, Sherali, HD and Shetty, CM (1993), *Nonlinear programming*, 2nd edition, Wiley, ISBN 0471557935, 9780471557937.

[5] Bazaraa, MS, Sherali, HD and Shetty, CM (2006), *Nonlinear programming*, John Wiley and Sons, ISBN 0471486000, 9780471486008.

[6] Belegundu, AD and Rajan, SD (1988), "A shape optimization approach based on natural design variables and shape functions," *Computer Methods in Applied Mechanics and Engineering*, 66 (1), 87–106, ISSN 0045-7825.

[7] Belitz, P and Bewley, T (2007), "Efficient derivative-free optimization," in "Decision and Control, 2007 46th IEEE Conference on," pages 5358–5363, ISBN 0191-2216.

[8] Berberian, SK (1994), *A first course in real analysis*, Springer, ISBN 0387942173, 9780387942179.

[9] Brandstatter, BR, Ring, W, Magele, C and Richter, KR (1998), "Shape design with great geometrical deformations using continuously moving finite element nodes," *Magnetics, IEEE Transactions on*, 34 (5), 2877–2880.

[10] Bugeda, G and Oate, E (February 1994), "A methodology for adaptive mesh refinement in optimum shape design problems," *Computing Systems in Engineering*, 5 (1), 91–102.

[11] Burmen, A and Tuma, T (2009), "Unconstrained derivative-free optimization by successive approximation," *Journal of Computational and Applied Mathematics*, 223 (1), 62–74, ISSN 0377-0427.

[12] Clarke, FH (1989), *Methods of dynamic and nonsmooth optimization*, number 57 in CBMS-NSF regional conference series in applied mathematics, Capital City Press, Montpelier, Vermont, USA, ISBN 0898712416, 9780898712414.

[13] Clarke, FH (1990), *Optimization and nonsmooth analysis*, number 5 in Canadian Mathematical Society series in mathematics, Wiley-Interscience, New York, NY, USA, ISBN 0898712564, 9780898712568.

[14] Conn, A, Scheinberg, K and Toint, P (October 1997), "Recent progress in unconstrained nonlinear optimization without derivatives," *Mathematical Programming, Series B*, 79 (1-3), 397–414.

[15] Cook, R, Malkus, D, Plesha, M and Witt, R (2002), *Concepts and applications of finite element analysis*, xvi, 719 p. : edition, Wiley, (New York), ISBN 624.171.

[16] Ding, Y (1986), "Shape optimization of structures: a literature survey," *Computers & Structures*, 24 (6), 985–1004.

[17] Dutta, J (December 2005), "Generalized derivatives and nonsmooth optimization, a finite dimensional tour," *TOP*, 13 (2), 185–279.

[18] Edelsbrunner, H (2001), *Geometry and topology for mesh generation*, Cambridge University Press, ISBN 0521793092, 9780521793094.

[19] Forrester, AI and Keane, AJ (2009), "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, 45 (1-3), 50–79, ISSN 0376-0421.

[20] Garcia, MJ and Gonzalez, CA (2004), "Shape optimisation of continuum structures via evolution strategies and fixed grid finite element analysis," *Structural and Multidisciplinary Optimization*, V26 (1), 92–98.

[21] Gould, N, Orban, D and Toint, P (2005), "Numerical methods for Large-Scale nonlinear optimization," *Acta Numerica*, 14 (-1), 299361.

[22] Groenwold, AA, Etman, LFP, Snyman, JA and Rooda, JE (2007), "Incomplete series expansion for function approximation," *Structural and Multidisciplinary Optimization*.

[23] Haftka, RT and Grandhi, RV (August 1986), "Structural shape optimization–A survey," *Computer Methods in Applied Mechanics and Engineering*, 57 (1), 91–106.

[24] Haftka, RT and Gurdal, Z (1991), *Elements of structural optimization*, volume 11 of *Solid Mechanics and its applications*, 3rd edition, Kluwer Academic Publishers, Dordrecht, the Netherlands.

[25] Herskovits, J, Dias, G, Santos, G and Soares, CM (October 2000), "Shape structural optimization with an interior point nonlinear programming algorithm," *Structural and Multidisciplinary Optimization*, 20 (2), 107–115.

[26] Hinton, E, Özakca, M and Rao, N (1991), "An integrated approach to structural shape optimization of linearly elastic structures. part II: shape definition and adaptivity," *Computing Systems in Engineering*, 2 (1), 41–56, ISSN 0956-0521, doi: 10.1016/0956-0521(91)90038-7.

[27] Hinton, E, Rao, NVR and Özakca, M (1991), "An integrated approach to structural shape optimization of linearly elastic structures. part i: General methodology," *Computing Systems in Engineering*, 2 (1), 27–39.

[28] Imam, MH (May 1982), "Three-dimensional shape optimization," *International Journal for Numerical Methods in Engineering*, 18 (5), 661–673.

[29] Joyce, DC (1971), "Survey of extrapolation processes in numerical analysis," *SIAM Review*, 13 (4), 435–490, ISSN 00361445.

[30] Kikuchi, N (April 1986), "Adaptive grid-design methods for finite element analysis," *Computer Methods in Applied Mechanics and Engineering*, 55 (1-2), 129–160.

[31] Kocks, UF (1976), "Laws for work-hardening and low-temperature creep." *J Eng Mater Technol Trans ASME*, 98 Ser H (1), 76–85.

[32] Kodiyalam, S and Thanedar, PB (1993), "Some practical aspects of shape optimization and its influence on intermediate mesh refinement," *Finite Elements in Analysis and Design*, 15 (2), 125–133.

[33] Kok, S, Beaudoin, AJ and Tortorelli, DA (April 2002), "On the development of stage IV hardening using a model based on the mechanical threshold," *Acta Materialia*, 50 (7), 1653–1667.

[34] Laporte, E and Tallec, PL (2002), *Numerical Methods in Sensitivity Analysis and Shape Optimization*, Modeling and Simulation in Science, Engineering and Technology, Birkhuser.

[35] Li, Q, Steven, GP, Querin, OM and Xie, YM (November 1999), "Evolutionary shape optimization for stress minimization," *Mechanics Research Communications*, 26 (6), 657–664, ISSN 0093-6413.

[36] Lui, D and Nocedal, J (August 1989), "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, 54 (1-3), 503–528.

[37] Martnez, R and Samartn, A (1991), "Two-dimensional mesh optimization in the finite element method," *Computers & Structures*, 40 (5), 1169–1175, ISSN 00457949.

[38] Mattheck, C and Burkhardt, S (May 1990), "A new method of structural shape optimization based on biological growth," *International Journal of Fatigue*, 12 (3), 185–190, ISSN 0142-1123.

[39] Miegroet, LV, Mos, N, Fleury, C and Duysinx, P (May 2005), "Generalized shape optimization based on the level set method," in "6th World Congresses of Structural and Multidisciplinary Optimization," pages 1–10.

[40] Olhoff, N, Rasmussen, J and Lund, E (1993), "A method of exact numerical differentiation for error elimination in finite element based semi-analytical shape sensitivity analysis," *Mechanics of Structures and Machines*, 21, 1–66.

[41] Peressini, AL, Sullivan, FE and Uhl, JJ (1988), *The mathematics of nonlinear programming*, Springer, ISBN 0387966145, 9780387966144.

[42] Persson, PO and Strang, G (2004), "A simple mesh generator in MATLAB," *SIAM Review*, 46 (2), 329–345.

[43] Potra, FA and Shi, Y (June 1995), "Efficient line search algorithm for unconstrained optimization," *Journal of Optimization Theory and Applications*, 85 (3), 677–704.

[44] Quapp, W (May 1996), "A gradient-only algorithm for tracing a reaction path uphill to the saddle of a potential energy surface," *Chemical Physics Letters*, 253 (3-4), 286–292.

[45] Rao, SS (2009), *Engineering Optimization: Theory and Practice*, John Wiley and Sons.

[46] Rardin, RL (August 1997), *Optimization in Operations Research*, Prentice Hall, ISBN 0023984155.

[47] Sacks, J, Welch, WJ, Mitchell, TJ and Wynn, HP (November 1989), "Design and analysis of computer experiments," *Statistical Science*, 4 (4), 409–423, ISSN 08834237, ArticleType: primary_article / Full publication date: Nov., 1989 / Copyright 1989 Institute of Mathematical Statistics.

[48] Schleupen, A, Maute, K and Ramm, E (July 2000), "Adaptive FE-procedures in shape optimization," *Structural and Multidisciplinary Optimization*, 19 (4), 282–302.

[49] Shor, NZ, Kiwiel, KC and Ruszcaynski, A (1985), *Minimization methods for non-differentiable functions*, Springer-Verlag New York, Inc., New York, NY, USA.

[50] Sienz, J and Hinton, E (July 1997), "Reliable structural optimization with error estimation, adaptivity and robust sensitivity analysis," *Computers & Structures*, 64 (1-4), 31–63, ISSN 0045-7949.

[51] Simpson, T, Toropov, V, Balabanov, V and Viana, F (September 2008), "Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come - or not," in "Proc. 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference," Victoria, British Columbia, Canada.

[52] Snyman, J and Hay, AM (July 2001), "The spherical quadratic steepest descent (SQSD) method for unconstrained minimization with no explicit line searches," *Computers and Mathematics with Applications*, 42 (1-2), 169–178.

[53] Snyman, JA (December 1982), "A new and dynamic method for unconstrained minimization," *Applied Mathematical Modelling*, 6 (6), 449–462.

[54] Snyman, JA (2005), "A gradient-only line search method for the conjugate gradient method applied to constrained optimization problems with severe noise in the objective function," *International Journal for Numerical Methods in Engineering*, 62 (1), 72–82.

[55] Snyman, JA (2005), *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, Applied Optimization, Vol. 97, 2nd edition, Springer-Verlag New York, Inc.

[56] Snyman, JA and Hay, AM (December 2002), "The Dynamic-Q optimization method: An alternative to SQP?" *Computers & Mathematics with Applications*, 44 (12), 1589–1598.

[57] Subramanian, G and Bose, JC (1982), "Convenient generation of stiffness matrices for the family of plane triangular elements," *Computers & Structures*, 15 (1), 85–89.

[58] Svanberg, K (2002), "A class of globally convergent optimization methods based on conservative convex separable approximations," *SIAM Journal on Optimization*, 12 (2), 555–573.

[59] Toropov, VV (1989), "Simulation approach to structural optimization," *Structural Optimization*, 1, 37–46.

[60] Turcke, D and Mcneice, GM (May 1974), "Guidelines for selecting finite element grids based on an optimization study," *Computers & Structures*, 4 (3), 499–519, ISSN 00457949.

[61] Van Keulen, F, Polynkine, AA and Toropov, VV (1997), "Shape optimization with adaptive mesh refinement: Target error selection strategies," *Eng Optim*, 28 (1-2), 95–125.

[62] Voce, E (1955), "A practical strain-hardening function," *Metallurgica*, 51, 219–226.

[63] Wall, WA, Frenzel, MA and Cyron, C (June 2008), "Isogeometric structural shape optimization," *Computer Methods in Applied Mechanics and Engineering*, 197 (33-40), 2976–2988, ISSN 0045-7825.

[64] Wallis, J (1685), *A treatise of algebra, both historical and practical*, London, London, England, published: Printed by J. Playford, for R. Davis.

[65] Wilke, DN, Kok, S and Groenwold, A (2010), "The application of gradient-only optimization methods for problems discretized using non-constant methods," *Structural and Multidisciplinary Optimization*, 40 (1), 433–451.

[66] Wilke, DN, Kok, S and Groenwold, AA (2006), "A quadratically convergent unstructured remeshing strategy for shape optimization," *International Journal for Numerical Methods in Engineering*, 65 (1), 1–17.

[67] Xie, Y and Steven, G (December 1993), "A simple evolutionary procedure for structural optimization," *Computers & Structures*, 49 (5), 885–896, ISSN 0045-7949.

[68] Zhang, L (2005), "A globally convergent bfgs method for nonconvex minimization without line searches," *Optimization Methods and Software*, 20, 737–747.

[69] Zienkiewicz, OC (2006), "The background of error estimation and adaptivity in finite element computations," *Adaptive Modeling and Simulation*, 195 (4-6), 207–213.

[70] Zienkiewicz, OC, Taylor, RL and Zhu, JZ (2005), *The finite element method*, Butterworth-Heinemann, ISBN 0750663200, 9780750663205.

[71] Zienkiewicz, OC and Zhu, JZ (1987), "A simple error estimator and adaptive procedure for practical engineerng analysis," *International Journal for Numerical Methods in Engineering*, 24 (2), 337–357.

Appendix

Gradient-only optimization requires the computation of accurate gradients. Various approaches are at our disposal to do this, e.g. direct differentiation and automatic differentiation. Finite difference schemes should be used with more caution; erroneous sign changes in particular are undesirable. The use of analytical or semi-analytical gradients renders gradient-only optimization computationally competitive. We do not explicitly give the sensitivities for the heat transfer problem; it is the easiest problem, and the approach is similar to the developments presented in Chapter 2.

Analytical sensitivities for the material identification study

In this study we compute the analytical sensitivities by direct differentiation of (3.44) and (3.45) with respect to the design variables i.e. θ_0, c, σ_{ys}, σ_4^0 and σ_y^0.

We start with θ_0. Since only (3.45) depends on θ_0, the sensitivity of σ_y^{i+1} w.r.t. θ_0 is given by

$$\begin{aligned} \frac{d\sigma_y^{i+1}}{d\theta_0} &= \frac{d\sigma_y^i}{d\theta_0} + \Big(1 - \frac{\sigma_y^i}{\sigma_{ys}} + \frac{\sigma_4^{i+1}}{\sigma_y^i}\Big)\Delta\epsilon_p^i \\ &+ \theta_0\Big(1 - \frac{d\sigma_y^i}{d\theta_0}\frac{1}{\sigma_{ys}} + \sigma_4^{i+1}\frac{d}{d\theta_0}(\frac{1}{\sigma_y^i})\Big)\Delta\epsilon_p^i. \end{aligned} \tag{6.1}$$

We note that (3.45) depends on (3.44), therefore both equations depend on c. In computing the sensitivity of σ_y^{i+1} w.r.t. c, we obtain

$$\begin{aligned} \frac{d\sigma_y^{i+1}}{dc} &= \frac{d\sigma_y^i}{dc} + \theta_0\Big(1 - \frac{d\sigma_y^i}{dc}\frac{1}{\sigma_{ys}} + \sigma_4^{i+1}\frac{d}{dc}(\frac{1}{\sigma_y^i}) \\ &+ \frac{d\sigma_4^{i+1}}{dc}\frac{1}{\sigma_y^i}\Big)\Delta\epsilon_p^i, \end{aligned} \tag{6.2}$$

and

$$\frac{d\sigma_4^{i+1}}{dc} = \frac{d\sigma_4^i}{dc} + \Big(\epsilon_p^{i+1} - \epsilon_p^i\Big). \tag{6.3}$$

Since only (3.45) depends on σ_{ys}, therefore the sensitivity of σ_y^{i+1} w.r.t. σ_{ys} is given by

$$\begin{aligned}\frac{d\sigma_y^{i+1}}{d\sigma_{ys}} &= \frac{\sigma_y^i}{d\sigma_{ys}} + \theta_0\Big(1 - \frac{d\sigma_y^i}{d\sigma_{ys}}\frac{1}{\sigma_{ys}} - \sigma_y^i\frac{d}{d\sigma_{ys}}(\frac{1}{\sigma_{ys}}) \\ &+ \sigma_4^{i+1}\frac{d}{d\sigma_{ys}}(\frac{1}{\sigma_y^i})\Big)\Delta\epsilon_p^i.\end{aligned} \tag{6.4}$$

Both (3.44) and (3.45) depend on σ_4^0, therefore the sensitivity of σ_y^{i+1} w.r.t. σ_4^0 is given by

$$\begin{aligned}\frac{d\sigma_y^{i+1}}{d\sigma_4^0} &= \frac{\sigma_y^i}{d\sigma_4^0} + \theta_0\Big(1 - \frac{d\sigma_y^i}{d\sigma_4^0}\frac{1}{\sigma_{ys}} + \frac{d\sigma_4^{i+1}}{d\sigma_4^0}\frac{1}{\sigma_y^i} \\ &+ \sigma_4^{i+1}\frac{d}{d\sigma_4^0}(\frac{1}{\sigma_y^i})\Big)\Delta\epsilon_p^i,\end{aligned} \tag{6.5}$$

whereas the sensitivity of σ_4^{i+1} w.r.t. σ_4^0 is given by

$$\frac{d\sigma_4^{i+1}}{d\sigma_4^0} = \frac{d\sigma_4^i}{d\sigma_4^0} = 1. \tag{6.6}$$

Since only (3.45) depends on σ_y^0, the sensitivity of σ_y^{i+1} w.r.t. σ_y^0 is given by

$$\frac{d\sigma_y^{i+1}}{d\sigma_y^0} = \frac{\sigma_y^i}{d\sigma_y^0} + \theta_0\Big(1 - \frac{d\sigma_y^i}{d\sigma_y^0}\frac{1}{\sigma_{ys}} + \sigma_4^{i+1}\frac{d}{d\sigma_y^0}(\frac{1}{\sigma_y^i})\Big)\Delta\epsilon_p^i. \tag{6.7}$$

Radial basis function mapping

The radial basis function $s(\boldsymbol{z})$ with $\boldsymbol{z} \in \mathbb{R}^2$ for the two dimensional case, is given by

$$s(\boldsymbol{z}) = \sum_{j=1}^{n_b} \alpha_j \phi(\|\boldsymbol{z} - \boldsymbol{\mathcal{X}}_j^{\partial\Omega}\|) + p(\boldsymbol{z}), \tag{6.8}$$

with p a polynomial, n_b the number of boundary nodes and ϕ a given basis function with respect to the norm $\|\boldsymbol{z}\|$. The coefficients α_j and the polynomial p are determined by the interpolation conditions

$$s(\boldsymbol{\mathcal{X}}_i^{\partial\Omega}) = \boldsymbol{d}_i^{\partial\Omega},\ i = 1, 2, \ldots, n_b \tag{6.9}$$

with $\boldsymbol{d}_i^{\partial\Omega}$ the displacement of the i^{th} boundary node. In addition it is required that

$$\sum_{j=1}^{n_b} \alpha_j q(\boldsymbol{\mathcal{X}}_j^{\partial\Omega}) = 0, \tag{6.10}$$

for all polynomials q with a degree less or equal than that of polynomial p. We rewrite (6.9) into a matrix form as

$$\boldsymbol{d}^{\partial\Omega} = \boldsymbol{M}\boldsymbol{\alpha} + \boldsymbol{P}\boldsymbol{\beta}, \tag{6.11}$$

where $\boldsymbol{M}$ is an $n_b \times n_b$ matrix with the i^{th} row and j^{th} column containing the evaluation of the basis function $\phi(\|\boldsymbol{\mathcal{X}}_i^{\partial\Omega} - \boldsymbol{\mathcal{X}}_j^{\partial\Omega}\|)$. For two dimensional interpolations $\boldsymbol{P}$ is an $n_b \times 3$

matrix with the i^{th} row given by $[1 \quad \mathcal{X}_{ix}^{\partial\Omega} \quad \mathcal{X}_{iy}^{\partial\Omega}]$, where $\mathcal{X}_{ix}^{\partial\Omega}$ and $\mathcal{X}_{iy}^{\partial\Omega}$ are respectively the x and y coordinates of the i^{th} boundary node. Similarly, (6.10) can be written in a matrix form

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{\alpha} = \mathbf{0}. \tag{6.12}$$

We therefore need to solve the two systems of linear equations (6.11) and (6.12). We start by rewriting (6.11) to obtain

$$\boldsymbol{\alpha} = \boldsymbol{M}^{-1}\boldsymbol{d}^{\partial\Omega} - \boldsymbol{M}^{-1}\boldsymbol{P}\boldsymbol{\beta}, \tag{6.13}$$

which we substitute into (6.12) to obtain

$$\boldsymbol{P}^{\mathrm{T}}(\boldsymbol{M}^{-1}\boldsymbol{d}^{\partial\Omega} - \boldsymbol{M}^{-1}\boldsymbol{P}\boldsymbol{\beta}) = \mathbf{0}.$$

We then solve for $\boldsymbol{\beta}$ from

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{P}\boldsymbol{\beta} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{M}^{-1}\boldsymbol{d}^{\partial\Omega}, \tag{6.14}$$

and $\boldsymbol{\alpha}$ from (6.13). After solving for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ the RBF $s(\boldsymbol{z})$ is defined and can be used to update the interior nodes $\boldsymbol{\mathcal{X}}^{\Omega}$ as the geometry changes. The boundary displacements $\boldsymbol{d}^{\partial\Omega}$ are obtained from the control variables $\boldsymbol{x}$ and piece-wise linear boundary interpolation.