



**Πανεπιστήμιο Θεσσαλίας**

**Τμήμα Μηχανολόγων Μηχανικών**

---

**Βελτιστοποίηση τοπολογίας κατασκευών με φορτία υπό  
αβεβαιότητα**

**Ευανθία Κάλλου**

Η παρούσα εργασία εκπονήθηκε στα πλαίσια απόκτησης του Διπλώματος  
Μηχανολόγου Μηχανικού

**Βόλος 2015**



**University of Thessaly**

**Department of Mechanical Engineering**

---

Robust topology optimization of structures

Evanthia Kallou

A thesis submitted in partial fulfillment of the requirements for the Diploma of  
Mechanical Engineering

**Volos 2015**



**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ  
ΒΙΒΛΙΟΘΗΚΗ & ΚΕΝΤΡΟ ΠΛΗΡΟΦΟΡΗΣΗΣ  
ΕΙΔΙΚΗ ΣΥΛΛΟΓΗ «ΓΚΡΙΖΑ ΒΙΒΛΙΟΓΡΑΦΙΑ»**

Αριθ. Εισ.: 14157/1  
Ημερ. Εισ.: 07-03-2017  
Δωρεά: Συγγραφέας  
Ταξιθετικός Κωδικός: ΠΤ – ΜΜ  
2015  
ΚΑΛ

© 2015 Evanthia Kallou

The approval of the thesis by the Mechanical Engineering department of University of Thessaly does not constitute acceptance of the author's beliefs (R. 5343/32 ar. 202 par. 2)

**Approved by the committee members:**

First Examiner (Supervisor): Professor Costas Papadimitriou

Department of Mechanical Engineering,  
University of Thessaly

Second Examiner:

Professor Spyros Karamanos

Department of Mechanical Engineering,  
University of Thessaly

Third Examiner:

Professor Gregory Haidemenopoulos

Department of Mechanical Engineering,  
University of Thessaly



## Contents

Table of Figures.....	2
Abstract .....	6
Acknowledgments .....	7
Chapter 1 - Introduction .....	8
1.1 Review in recent and relevant literature .....	8
1.2 Objectives and Thesis Outline.....	11
Chapter 2 - The static load case.....	12
2.1 Theoretical background of topology optimization .....	12
2.2 Implementation in Matlab .....	17
2.3 Results and Discussion .....	22
Chapter 3 - The robust load case .....	73
3.1 Theoretical background for robust topology optimization .....	73
3.2 Implementation in Matlab .....	77
3.3 Results and discussion .....	79
Chapter 4 - Concluding remarks.....	90
Bibliography .....	92
Appendix .....	95

## Table of Figures

Fig. 2.1: Optimal design for volume fraction $\text{volfrac}=0.1$ , by using the prompt line $\text{top1}(20, 40, 0.1, 4, 2.5)$ .....	23
Fig. 2.2: Optimal design for volume fraction $\text{volfrac}=0.1$ , by using the prompt line $\text{top1}(30, 60, 0.1, 4, 2.5)$ .....	23
Fig. 2. 3: Optimal design for volume fraction $\text{volfrac}=0.1$ , by using the prompt line $\text{top1}(50, 100, 0.1, 4, 2.5)$ .....	24
Fig. 2.4: Optimal design for volume fraction $\text{volfrac}=0.1$ , by using the prompt line $\text{top1}(100, 200, 0.1, 4, 2.5)$ .....	25
Fig. 2. 5: Optimal design for volume fraction $\text{volfrac}=0.3$ , by using the prompt line $\text{top1}(20, 40, 0.3, 4, 2.5)$ .....	26
Fig. 2. 6: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(20, 40, 0.4, 4, 2.5)$ .....	27
Fig. 2.7: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(30, 60, 0.4, 4, 2.5)$ .....	27
Fig. 2.8: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(50, 100, 0.4, 4, 2.5)$ .....	28
Fig. 2.9: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(20, 40, 0.7, 4, 2.5)$ .....	29
Fig. 2.10: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(30, 60, 0.7, 4, 2.5)$ .....	29
Fig. 2.11: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(50, 100, 0.7, 4, 2.5)$ .....	30
Fig. 2.12: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(100, 200, 0.7, 4, 2.5)$ .....	30
Fig. 2.13: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(50, 25, 0.2, 4, 2.5)$ .....	31
Fig. 2.14: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(100, 50, 0.2, 4, 2.5)$ .....	32
Fig. 2.15: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(50, 25, 0.5, 4, 2.5)$ .....	32
Fig. 2.16: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(100, 50, 0.5, 4, 2.5)$ .....	33
Fig. 2. 17: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(50, 25, 0.7, 4, 2.5)$ .....	33
Fig. 2.18: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(100, 50, 0.7, 4, 2.5)$ .....	34
Fig. 2. 19: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(60, 20, 0.2, 4, 2.5)$ .....	35
Fig. 2.20: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(120, 40, 0.2, 4, 2.5)$ .....	35
Fig. 2.21: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(300, 100, 0.2, 4, 2.5)$ .....	36

Fig. 2.22: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(60, 20, 0.5, 4, 2.5)$ .....	36
Fig. 2. 23: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(120, 40, 0.5, 4, 2.5)$ .....	37
Fig. 2. 24: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(300, 100, 0.5, 4, 2.5)$ .....	37
Fig. 2.25: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(60, 20, 0.7, 4, 2.5)$ .....	38
Fig. 2.26: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(120, 40, 0.7, 4, 2.5)$ .....	38
Fig. 2. 27: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(300, 100, 0.7, 4, 2.5)$ .....	39
Fig. 2.28: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(30, 60, 0.2, 4, 2.5)$ .....	40
Fig. 2.29: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(75, 150, 0.2, 4, 2.5)$ .....	41
Fig. 2.30: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(30, 60, 0.5, 4, 2.5)$ .....	42
Fig. 2.31: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(75, 150, 0.5, 4, 2.5)$ .....	42
Fig. 2. 32: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(30, 60, 0.7, 4, 2.5)$ .....	43
Fig. 2.33: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(75, 150, 0.7, 4, 2.5)$ .....	43
Fig. 2.34: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(30, 150, 0.2, 4, 2.5)$ .....	44
Fig. 2.35: Optimal design for volume fraction $\text{volfrac}=0.2$ , by using the prompt line $\text{top1}(40, 200, 0.2, 4, 2.5)$ .....	45
Fig. 2.36: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(30, 150, 0.4, 4, 2.5)$ .....	45
Fig. 2.37: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(40, 200, 0.4, 4, 2.5)$ .....	46
Fig. 2.38: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(30, 150, 0.7, 4, 2.5)$ .....	46
Fig. 2.39: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(40, 200, 0.7, 4, 2.5)$ .....	47
Fig. 2.40: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(50, 25, 0.4, 4, 2.5)$ .....	48
Fig. 2.41: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(150, 75, 0.4, 4, 2.5)$ .....	48
Fig. 2.42: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(50, 25, 0.5, 4, 2.5)$ .....	49
Fig. 2. 43: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(150, 75, 0.5, 4, 2.5)$ .....	49

Fig. 2.44: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(20, 40, 0.4, 4, 2.5)$ .....	50
Fig. 2.45: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(30, 60, 0.4, 4, 2.5)$ .....	51
Fig. 2.46: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(40, 80, 0.4, 4, 2.5)$ .....	51
Fig. 2.47: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(20, 40, 0.5, 4, 2.5)$ .....	52
Fig. 2.48: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(30, 60, 0.5, 4, 2.5)$ .....	52
Fig. 2. 49: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(40, 80, 0.5, 4, 2.5)$ .....	53
Fig. 2.50: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(20, 40, 0.7, 4, 2.5)$ .....	53
Fig. 2.51: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(30, 60, 0.7, 4, 2.5)$ .....	54
Fig. 2. 52: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(40, 80, 0.7, 4, 2.5)$ .....	54
Fig. 2.53: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(20, 100, 0.5, 4, 2.5)$ .....	55
Fig. 2.54: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(30, 150, 0.5, 4, 2.5)$ .....	56
Fig. 2.55: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(20, 100, 0.7, 4, 2.5)$ .....	56
Fig. 2.56: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(30, 150, 0.7, 4, 2.5)$ .....	57
Fig. 2.57: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(40, 20, 0.4, 4, 2.5)$ .....	58
Fig. 2.58: Optimal design for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(100, 50, 0.4, 4, 2.5)$ .....	58
Fig. 2.59: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(40, 20, 0.5, 4, 2.5)$ .....	59
Fig. 2.60: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(100, 50, 0.5, 4, 2.5)$ .....	59
Fig. 2.61: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(40, 20, 0.7, 4, 2.5)$ .....	60
Fig. 2.62: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(100, 50, 0.7, 4, 2.5)$ .....	60
Fig. 2.63: Optimal design for volume fraction $\text{volfrac}=0.3$ , by using the prompt line $\text{top1}(50, 100, 0.3, 4, 2.5)$ .....	61
Fig. 2.64: Optimal design for volume fraction $\text{volfrac}=0.3$ , by using the prompt line $\text{top1}(100, 200, 0.3, 4, 2.5)$ .....	62
Fig. 2.65: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(50, 100, 0.5, 4, 2.5)$ .....	62

Fig. 2.66: Optimal design for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(100, 200, 0.5, 4, 2.5)$ .....	63
Fig. 2.67: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(50, 100, 0.7, 4, 2.5)$ .....	63
Fig. 2.68: Optimal design for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(100, 200, 0.7, 4, 2.5)$ .....	64
Fig. 2.69: Optimal design (fixed at both ends) for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(60, 20, 0.7, 4, 2.5)$ .....	66
Fig. 2.70: Optimal design (fixed at both ends) for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(60, 20, 0.7, 4.5, 3.5)$ .....	66
Fig. 2.71: Optimal design (scrolling and joint) for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(100, 50, 0.7, 4, 2.5)$ .....	67
Fig. 2.72: Optimal design (scrolling and joint) for volume fraction $\text{volfrac}=0.7$ , by using the prompt line $\text{top1}(100, 50, 0.7, 4.5, 2.5)$ .....	68
Fig. 2.73: Optimal design (fixed left end) for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(150, 75, 0.5, 4, 2.5)$ .....	69
Fig. 2.74: Optimal design (fixed left end) for volume fraction $\text{volfrac}=0.5$ , by using the prompt line $\text{top1}(150, 75, 0.5, 7, 2.5)$ . .....	70
Fig. 2.75: Optimal design (scrolling and joint) for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(40, 20, 0.4, 4, 2.5)$ .....	71
Fig. 2.76: Optimal design (scrolling and joint) for volume fraction $\text{volfrac}=0.4$ , by using the prompt line $\text{top1}(40, 20, 0.4, 4.5, 3.5)$ .....	72
Fig. 3.1: Optimal design for the static case problem, by using the prompt line $\text{top1}(30, 60, 0.4, 4, 2.5)$ .....	79
Fig. 3.2: Optimal design for the uncertain case problem, by using the prompt line $\text{top2}(30, 60, 0.4, 4, 2.5, 0.5, -1, 0.3)$ .....	80
Fig. 3.3: Optimal design for the uncertain case problem, by using the prompt line $\text{top2}(30, 60, 0.4, 4, 2.5, 0.6, -1, 0.2)$ .....	80
Fig. 3.4: Optimal design for the static case problem, by using the prompt line $\text{top1}(30, 60, 0.7, 4, 2.5)$ .....	82
Fig. 3.5: Optimal design for the uncertainty problem, by using the prompt line $\text{top2}(30, 60, 0.7, 4, 2.5, 0.8, -1, 0.2)$ .....	82
Fig. 3.6: Optimal design for the static case problem, by using the prompt line $\text{top1}(100, 50, 0.5, 4, 2.5)$ .....	83
Fig. 3.7: Optimal design for the uncertainty problem, by using the prompt line $\text{top2}(100, 50, 0.5, 4, 2.5, 0.3, -1, 0.35)$ .....	84
Fig. 3.8: Optimal design for the static case problem, by using the prompt line $\text{top1}(30, 60, 0.7, 4, 2.5)$ .....	85
Fig. 3.9: Optimal design for the uncertainty problem, by using the prompt line $\text{run\_top2n}(30, 60, 0.7, 4, 2.5, 0.3, -1, 3, 0.3, 0.9)$ .....	85
Fig. 3.10: Optimal design for the uncertainty problem, by using the prompt line $\text{run\_top2n}(30, 60, 0.7, 4, 2.5, 0.3, -1, 3, 0.35, 1.05)$ .....	86
Fig. 3.11: Optimal design for the static case problem, by using the prompt line $\text{top1}(25, 50, 0.4, 4, 2.5)$ .....	87
Fig. 3.12: Optimal design for the uncertainty problem, by using the prompt line $\text{run\_top2n}(25, 50, 0.4, 4, 2.5, 0.3, -1, 3, 0.35, 1.05)$ .....	87



## Abstract

Topology optimization is a methodology that provides the optimal layout of a structure within a predefined design domain. The objective of this thesis is to present a framework for topology optimization of structures for various boundary conditions and to also expand to the case of load robust topology optimization. Three different Matlab cases implemented in Matlab code, are presented, one for the static load case and the two others for uncertainty in one and two loads, respectively. These three different code variations derive from the original 99-line code by Sigmund (1999) for topology optimization of 2-D structures.

For the deterministic case, different boundary conditions were examined and the values of the variables **penal** and **rmin** were studied, as well as the number of elements. **Rmin** is the filter size that is divided by the size of each element and **penal** is the penalization power from the SIMP methodology which is permissible for  $p \geq 3$  (Sigmund, Bendsoe, 1999). It was shown that for a finer mesh, the optimal structure becomes more complex and with less gray areas, meaning a better distribution of material. Moreover, By increasing the **penal** variable above the value 4.5 the gray areas are not affected. However, if increased too much, the topology may change and prove inaccurate. The ideal value for **penal** is around 4. The ideal value for **rmin** is 2.5. For a larger value, there appear more gray areas.

For the load uncertainty case, two codes were built for the topology optimization, also based on the original case of Sigmund. Code **top2** is built for the case of one-load uncertainty and code **run\_top2n** is built for uncertainty in two loads. For the uncertainty in one load only, the topology does not change in comparison to the static case problem, especially for the symmetrical problem for fixed ends at both sides and load in the middle. For more apparent alterations, a more asymmetrical model was required and so the two load problem was inserted. The results were indeed more obvious and the alterations between the deterministic and the uncertainty problem were mainly observed at the point where the forces were applied.

## **Acknowledgments**

I would like to express my gratitude to the supervisor of my thesis, Professor Costas Papadimitriou for his guidance and support, as well as to Professor Spyros Karamanos and Professor Gregory Haidemenopoulos for their useful suggestions. I would also like to thank them for the knowledge I gained by attending their courses. In addition, I want to offer a sincere thank you to Mr Dimitris Papadimitriou and to all the members of the Systems Dynamics Laboratory for their collaboration to this thesis. Last but not least, I would like to thank my parents, George and Stella and my brother Andreas, for their love and understanding, as well as to all my friends for their support.

# Chapter 1

## Introduction

### 1.1 Review in recent and relevant literature

The design optimization of structures is a subject which has enabled engineers to discover solutions for engineering problems, such as optimizing the size and shape of a structure within a given domain. Over the years, there have been multiple approaches to this subject, with the one of topology optimization constantly gaining ground. But what exactly is topology optimization? Topology optimization is a relatively new methodology which has been used widely over the last decade in the aerospace, mechanical and material fields, with more recent applications in the design of Micro-Electro and Mechanical Systems, also known as MEMS. Its application has achieved great minimization in material use and consequently, in cost, since it is computer-aided and its goal is to optimize the distribution of a limited amount of material in a specific domain. This is the reason that many publications have been made over the years over this matter.

For the static case, the pylon of topology optimization is the work of Sigmund (1999) who presented ‘A 99 line topology optimization code written in Matlab’ a code which enables engineers to calculate an optimal topology when boundary conditions and volume fraction of the material are given. This code, as well as an improved one using 88 lines of code in Matlab (Sigmund et al., 2010) have enabled the visualization of the optimal design and are based on the optimization through compliance minimization on statically loaded structures. The 88-line code has also the extension of a density filter and was described by the authors as more efficient than the original 99-line code developed by Sigmund. In the 99-line code (Sigmund, 1999), the Solid Isotropic Material with Penalization (SIMP) approach was used. Otherwise,



this methodology is called ‘power-law approach’ (Bendsoe, 1989; Zhou and Rozvany, 1991; Mlejnek, 1992). The material properties are considered constant and the densities related to each element are the variables. The theory of the optimization is described on isotropic and anisotropic materials, as well as its extensions and applications and the topology design of truss structures, in the book ‘Topology Optimization: Theory, Methods and Applications’ (Sigmund and Bendsoe, 2003).

Matlab was used for the realization of these codes, because it provides excellent tools and great efficiency, especially in finite element analysis which is used for the calculation of the optimization code. Such examples are a finite element code for the solution of elliptic problems with mixed boundary conditions on unstructured grids (Alberty et al, 1999), a topology optimization code for compliant mechanism design and heat conduction problems (Bendsoe and Sigmund, 2003), a discrete level-set topology optimization code (Challis, 2010), a combined shape and topology optimization methodology for 3D structures (Christiansen et al., 2014). Recent review papers have also demonstrated that, for compliant structures under large displacements or large rotation, geometrical non-linear analysis is implemented into the original topology optimization approach. However, when non-linearity appears, the standard finite element method proves inadequate, and so the use of an element-free Galerkin method (EFG) is required (Wang et al., 2015). This method has also been used for the topology design of cracked structures (Shobeiri, 2015).

Another method for the development of the topology optimization of compliant structures is the one proposed by Huang et al, (2014), which is based on the flexibility of the components of such structures and is called BESO (bi-directional evolutionary structural optimization). Finally, with the available computing resources, the investigation of the material microstructure composition has permitted the solution of both the macroscopic and the microscopic topology problem via homogenization processes such as  $FE^2$  (Xia and Breitkopf, 2014). This approach has been developed ‘*in order to assess the macroscopic influence of microscopic heterogeneities*’, as quoted by Xia and Breitkopf, 2014. None of the above methods have, however, taken into consideration the uncertainty from the imposed excitation.

Throughout the years, and based on the theories mentioned above, the need for the application of topology optimization under uncertainty in load and/or element density

has arisen. Carrasco et al. (2014) proposed a uncertainty model for topology optimization, where they introduced a variance-expected compliance model where the problem becomes nonlinear and thus its complexity reduces. Another work presents a uncertainty topology optimization of structures with imperfect geometry, where the solution is based on geometric linear analysis through a Total Lagrangian finite element methodology (Jansen et al. 2014). In this problem, the Monte Carlo sampling method has been used to estimate the uncertainty quantities. Their approach allowed the solution of 2D and 3D design problems with loads with dependent or independent perturbations. A very important work upon uncertainty and reliability-based topology optimization has been presented by Guest and Igusa (2008), where uncertainties in loading and nodal locations have been implemented into calculations and then resolved, using the adjoint method.

## 1.2 Objectives and Thesis Outline

Despite all the important developments made on the comprehension and solution of topology optimization problems, there are still many topics worth working on. In this thesis work, the subjects that will be studied are:

- The topology optimization of a beam under deterministic load and different boundary conditions with the use of the SIMP method.
- The implementation of Von Mises stress criteria for the topology optimal structure.
- The robust topology optimization of a cantilever beam under uncertain load.

The research work presented in this thesis is organized as follows:

Chapter 2 introduces the static load case problem and demonstrates the topology optimal structures for different boundary conditions and load applications. The stresses for the optimal structure are also calculated and compared to the yield strength. The code is written in Matlab and is based on the 99-line code by Sigmund (1999) that uses the finite element method, a sensitivity filter and an optimality criterion. The effect of the number of elements, as well as the volume fraction, the penalty and the dimensions of the beam are discussed.

In Chapter 3, a methodology for design optimization of structures under uncertain load is presented for the example of the cantilever beam. The code presented in Chapter 2 is modified in order to include the load uncertainty according to the robust methodology described in the work by Papadimitriou and Papadimitriou (2015). Different standard deviation values are studied and discussed.

Finally, some concluding remarks are presented in Chapter 4 regarding the results obtained from the methodologies in Chapters 2 and 3.

# Chapter 2

## The static load case

### 2.1 Theoretical background of topology optimization

Topology optimization is a methodology that provides the optimal layout of a structure within a predefined design domain. This is achieved by mathematically approaching a structural problem. More specifically in our case, this is done for given load and boundary conditions. In this Section, parts of the Sigmund (1999) methodology are used for complicity.

Since the problem at hand is that of a beam with different boundary conditions, the design domain is assumed to be rectangular with square finite elements discretization. For simplicity reasons, the numbering of nodes and elements begins from the upper left node and continues column by column. Following the SIMP approach for compliance minimization that was referred in Chapter 1, the topology optimization problem, according to Sigmund (1999) is expressed below:

$$\begin{aligned} \min_{\mathbf{x}}: \mathbf{c}(\mathbf{x}) &= \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_e \mathbf{u}_e \\ \text{subject to: } \frac{V(\mathbf{x})}{V_0} &= f \\ &: \mathbf{K} \mathbf{U} = \mathbf{F} \\ &: 0 < x_{\min} \leq x \leq 1 \end{aligned}$$

At the expression above,  $\mathbf{c}$  is the compliance,  $\mathbf{x}$  is the vector of the element densities (i.e. design variables),  $\mathbf{K}$  is the global stiffness matrix,  $\mathbf{U}$  and  $\mathbf{F}$  are the global displacement and force vectors respectively,  $\mathbf{u}_e$  is the element displacement vector,  $\mathbf{k}_e$  is the element stiffness matrix,  $N$  is the number of elements,  $V(\mathbf{x})$  and  $V_0$  are the material volume and the design domain volume respectively and  $f$  is the volume fraction. Last but not least,  $p$  is the penalization power that according to a work by Bendsoe and Sigmund (1999) must be equal to or larger than 3 in order for the SIMP approach to be permissible.

Next step to the solution of the optimization problem expressed above is the choice of an approach. Here, the optimality criteria (OC) method was chosen. Others methods include the Method of Moving Asymptotes (Svanberg, 1987) and Sequential Linear Programming (SLP). Therefore, according to the work of Bendsoe (1995) and later Sigmund (1999), the update for the density of each material becomes:

$$\begin{aligned} x_e^{new} &= \{ \max(x_{min}, x_e - m), \text{ if } x_e B_e^n \leq \max(x_{min}, x_e - m) \}, \\ &= \{ x_e B_e^n, \text{ if } \max(x_{min}, x_e - m) < x_e B_e^n < \min(1, x_e + m) \}, \\ &= \{ \min(1, x_e + m), \text{ if } \min(1, x_e + m) \leq x_e B_e^n \} \end{aligned}$$

Here,  $m$  is a positive move-limit,  $n=0.5$  is a damping coefficient and  $B_e$  is found from the optimality condition and is expressed as:

$$B_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}}$$

where  $\lambda$  is a Lagrangian multiplier of the optimization problem.

The sensitivity of the objective function is expressed by its derivative as:

$$\frac{\partial c}{\partial x_e} = -p(x_e)^{p-1} \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$$

Finally, in order to ensure the mesh-independency of the optimal designs and also to eliminate potential ‘checkerboard patterns’ (Diaz and Sigmund, 1995; Jog and Haber, 1996; Sigmund and Peterson, 1998), a mesh-independency filter is required:

$$\widehat{\frac{\partial c}{\partial x_e}} = \frac{1}{x_e \sum_{g=1}^N \widehat{H}_g} \sum_{g=1}^N \widehat{H}_g x_g \frac{\partial c}{\partial x_g}$$

where the convolution operator (weight factor)  $\widehat{H}_g$  is written as:

$$\widehat{H}_g = r_{min} - \text{dist}(e, g), \{g \in N | \text{dist}(e, g) \leq r_{min}\}, e=1, \dots, N.$$

From its expression, it is apparent that as the distance from element  $g$  increases,  $\widehat{H}_g$  becomes smaller and reaches the value zero when it leaves the filtering region. The operator  $\text{dist}(e, g)$  is defined as the distance between the center of element  $e$  and the center of element  $g$ .

The problem expressed above by Sigmund (1999) provides the optimal topology for the given structure. However, it does not provide any information regarding the stresses developed at the optimized structure. In the work that follows an approach to the stress and strain calculation will be presented.

For biaxial (plane) stress, the stress matrix for each element is:

$$[\sigma] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & 0 \\ \sigma_{yx} & \sigma_{yy} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

where  $\sigma_{xx}$  is the x-axial stress,  $\sigma_{yy}$  is the y-axial stress and  $\sigma_{xy} = \sigma_{yx}$  is the shear stress. In this problem the material is assumed to be linearly elastic and isotropic, so the strains and the stresses can be calculated from the following equations:

Strains	Stresses
$e_{xx} = \frac{\partial u_x}{\partial x}$	$\sigma_{xx} = \frac{E}{(1+\nu)(1-\nu)}(e_{xx} + \nu e_{yy})$
$e_{yy} = \frac{\partial u_y}{\partial y}$	$\sigma_{yy} = \frac{E}{(1+\nu)(1-\nu)}(e_{yy} + \nu e_{xx})$
$e_{xy} = e_{yx} = \frac{1}{2} \left( \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right)$	$\sigma_{xy} = \frac{E}{(1+\nu)} e_{xy}$

$u_x$  and  $u_y$  are the displacements calculated for each node for the x and y axis, respectively. More specifically,  $u_x$  for node  $i$  is found on the  $2i-1$  position on the global displacement matrix  $U$  and  $u_y$  is found on the  $2i$  position. From the above it is apparent that the strains for each element must be found by the displacements of all the nodes of the corresponding element. From the finite element method (FEM), the Galerkin equations are used to describe the position of element at a local axis system ( $\xi, \eta$ ). For an square element, these equations are the following:

$$N_1 = \frac{1}{4}(1 - \xi)(1 + \eta)$$

$$N_2 = \frac{1}{4}(1 + \xi)(1 + \eta)$$

$$N_3 = \frac{1}{4}(1 + \xi)(1 - \eta)$$

$$N_4 = \frac{1}{4}(1 - \xi)(1 - \eta)$$

Now, the total displacement on the x and y-axis, respectively, can be calculated through Galerkin as:

$$u_q = \sum_{i=1}^N N_i * u_{qi} \text{ and } u_n = \sum_{i=1}^N N_i * u_{ni}$$

In order to find the strain, it is simply needed to find the derivative of each Galerkin equation and multiply it by its counterpart displacement:

$$e_{xx} = \sum_{i=1}^N \frac{\partial N_i}{\partial q} * u_{qi}$$

$$e_{yy} = \sum_{i=1}^N \frac{\partial N_i}{\partial n} * u_{ni}$$

$$e_{xy} = \frac{1}{2} \left( \sum_{i=1}^N \frac{\partial N_i}{\partial q} * u_{ni} + \sum_{i=1}^N \frac{\partial N_i}{\partial n} * u_{qi} \right)$$

This way, the stresses can be directly found by the equations on the matrix shown above. For each element, according to the Von Mises criterion, an equivalent stress  $\sigma_e$  must be calculated and compared to the yield strength of the material  $\sigma_y$ . The formula for the equivalent stress is:

$$\sigma_e = \sqrt{(\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\sigma_{xy}^2)}$$

The equivalent stress must be equal or smaller than the yield strength of the material in order to avoid failure of the element  $i$ .

In the following Chapter, the methodology expressed above for topology optimization and calculation of strains and stresses will be implemented into the Matlab code.



## 2.2 Implementation in Matlab

The methodology explained above is implemented in a topology optimization code and will be explained briefly in this Chapter. This code is built in Matlab according to the one by Sigmund (1999) and is called by the prompt line

**top1(nelx, nely, volfrac, penal, rmin)**

where **nelx**, **nely** are the number of elements in the x-axis and y-axis, respectively, **volfrac** is the volume fraction used to distribute the material and **rmin** is the filter size that is divided by the size of each element. Finally, **penal** is the penalization power from the SIMP methodology which is permissible for  $p \geq 3$  (Sigmund, Bendsoe, 1999).

The code 'top1' contains many subroutines that are called in the main program and are fully presented in the Appendix. First of all, in the first lines of the main program, there is a line that determines the even distribution of material in the design domain, always according to the volume fraction that was inserted by the user. Next, there is the initialization of a loop that relies on the value of the variable **change**. This variable starts with the value 1 and changes (as the name implies) inside the loop.

After that, the subroutine [U] is called which solves the problem by using the finite element method. The FEM subroutine calls the [KE] subroutine, which calculates the values of the element stiffness matrix. After the element stiffness matrix has a value, the FEM problem is solved, depending on the boundary conditions that are required by the problem. In the code of Sigmund (1999), the boundary conditions studied are those for the example of the cantilever beam with load application on the down right corner and also the example of a beam with scrolling at the left side and at the down right corner and load on the upper left corner. In this work, different boundary conditions were tested in the Matlab code. These boundary conditions for the load and displacement are presented on the matrix below. The displacement boundary conditions are implemented in the code by the variable **fixeddofs**. In the code presented in the Appendix, all the boundary conditions are written and, according to the type of beam, the user can remove the '%' comment Figure and activate the condition preferred. By adding the '%' Figure, all the other boundary conditions become inactive.

Type of boundary	Load Condition (F)	Displacement Condition (fixeddofs=)
Fixed ends at both sides	At the middle of the beam: $F((nely+1)*nelx+2,1) = -1$	$\text{union}([1:2*(nely+1)], [2*(nely+1)*nelx-1:2*(nelx+1)*(nely+1)])$
Fixed end at the left side	At the upper right corner: $F(2*(nely+1)*(nelx+1)-2*nely,1)=-1$	$[1:2*(nely+1)]$
Scrolling at the lower left corner and joint at the lower right corner	At the middle of the beam (upper side): $F((nely+1)*nelx+2,1) = -1$	$\text{union}([2*(nely+1)], [2*(nelx+1)*(nely+1)-1, 2*(nelx+1)*(nely+1)])$
Fixed end at the left side	Uniformly distributed at the upper side of the beam: $F([2*(nely+1)+2:2*(nely+1):2*(nely+1)*(nelx+1)-2*nely],1)=-1/nelx$	$[1:2*(nely+1)]$
Fixed end at the left side	At the upper right corner: $F(2*(nely+1)*(nelx+1)-2*nely,1)=-1$ <b>and</b> At the middle of the beam (lower side): $F((nely+1)*(nelx+2),1) = -1$	$[1:2*(nely+1)]$

The [KE] subroutine is also called right after [U] and is used in the objective function **c** and sensitivity analysis part. For the sensitivity analysis part, the derivative of the objective function **dc** is calculated for each loop. Then, for the filtering of sensitivities, the subroutine [dc] is called for each derivative **dc**. The design is updated according to the optimality criteria method by the subroutine OC. Finally, the results and the densities are printed for each loop.

All the above concerned the topology optimization problem. Now the strain and stress calculation part of the code will be explained. By following the classic theory of Mechanics of Structures and by using the FEM theory, the following lines of code were implemented into the original code (Sigmund, 1999):

## Stress and Strain calculation algorithm

```

%%%STRESS AND STRAIN CALCULATION%%%%%%%%
Sy=1.0; %%Yield Strength
nu=0.3; %%For Steel
E = 1;
Sxx=0; Sxy=0; Syy=0;
exx=0; exy=0; eyy=0;
for ely=1:nely
    for elx=1:nelx
        if x(ely,elx)>=0.500
            n1=(nely+1)*(elx-1)+ely;
            n2=(nely+1)*elx+ely;
            exx=-(1/4)*(2)*U(2*n1-1,:)+(1/4)*(2)* U(2*n2-1,:)+(1/4)*(2)*
U(2*n2+1,:)-(1/4)*(2)* U(2*n1+1,:);
            eyy=(1/4)*(2)*U(2*n1,:)+(1/4)*(2)*U(2*n2,:)-
(1/4)*(2)*U(2*n2+2,:)-(1/4)*(2)*U(2*n1+2,:);
            exy=(1/2)*((1/4)*(2)*U(2*n1-1,:)+(1/4)*(2)*U(2*n2-1,:)-
(1/4)*(2)*U(2*n2+1,:)-(1/4)*(2)*U(2*n1+1,:)-(1/4)
*(2)*U(2*n1,:)+(1/4)*(2)*U(2*n2,:)+(1/4)*(2)*U(2*n2+2,:)-
(1/4)*(2)*U(2*n1+2,:));
            Sxx=(E/(1-(nu)^2))*(exx+nu*eyy);
            Syy=(E/(1-(nu)^2))*(eyy+nu*exx);
            Sxy=(E/(1+nu))*exy;
            Se=sqrt(Sxx^2-Sxx*Syy+Syy^2+3*Sxy^2);
            %%%Von Mises Criteria
            if Se>Sy
                display('Failure for element' )
                display('Ely=')
                display(ely)
                display('Elx=')
                display(elx)
                display('Stress Se=')
                display(Se)
            else
                display('Stress under yield for element' )
                display('Ely=')
                display(ely)
                display('Elx=')
                display(elx)
                display('Stress Se=')
                display(Se)
            end
        end
    end
end
end
end

```

From the basic code of Sigmund (1999), the Young's modulus **E** was normalized and is set equal to 1 and the Poisson's ratio **nu** is set to 0.3 that is the value for steels. For the Von Mises criterion, the yield strength **Sy** is set to 1.0. The constants **E**, **nu** and **Sy** can be altered by the user. The strain variables are **exx**, **exy** and **eyy** and the stress variables are **Sxx**, **Sxy** and **Syy** and they are all initialized outside the loops. The indexes **elx** and **ely** are used in order to check if the corresponding element has density equal or larger than 0.5. Through the 'if' condition, the strain and stresses will be calculated only for elements with density equal or larger than 0.5. This way, the computational costs become much smaller.

After the variables **exx**, **exy**, **eyy** and **Sxx**, **Sxy**, **Syy** have been calculated according to the methodology explained in Chapter 2.1 (FE and Mechanics of Materials methods), the equivalent stress **Se** is found and then compared to the yield strength **Sy**. Then, the counterpart message appears on screen, depending on the value of **Se**.

## 2.3 Results and Discussion

In the previous Sections of Chapter 2, there was a brief description and also the additions to of the original code that was developed by Sigmund (1999). More specifically, in Section 2.1, the methodology of both the topology and the stress and strain problem was presented. In Section 2.2, this methodology was implemented into a Matlab code and also various boundary conditions were studied. The results that were shown by the **top1** code refer to the topology optimization part and specifically, to the different boundary conditions. The significance of the ratio  $h/L$  of the beam is thoroughly studied, where  $h$  is the height of the beam in the 2D case and  $L$  is its length. It is proved that, for square finite elements,  $Dx=Dy$ , where  $Dx$ ,  $Dy$  is the distance between two successive nodes on the horizontal and vertical axis, respectively. The total beam length  $L$  is in this case equal to  $nelx*Dx$  and the total height  $h$  is  $nely*Dy$ . This way, the  $h/L$  ratio becomes equal to the  $nely/nelx$  ratio. This means that by alternating the ratio of elements one also alternates the dimensions ratio.

Finally, the importance of the variables **rmin** and **penal** to the optimal design is also underlined, as well as of the number of elements. These various cases are presented below:

### 1. Fixed ends at both sides with load applied in the middle of the upper side of the beam:

As referred to in Section 2.2, the prompt line to call the code is **top1(nelx, nely, volfrac, penal, rmin)**.

- For  $h/L = nely/nelx = 2$ ,

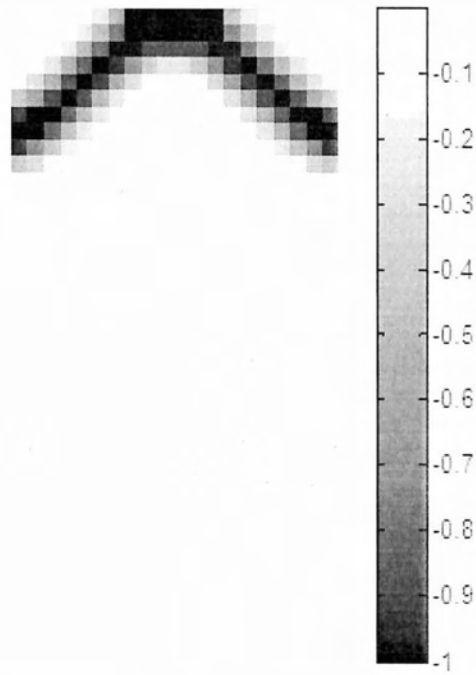


Fig. 2.1: Optimal design for volume fraction  $\text{volfrac}=0.1$ , by using the prompt line `top1(20,40,0.1,4,2.5)`

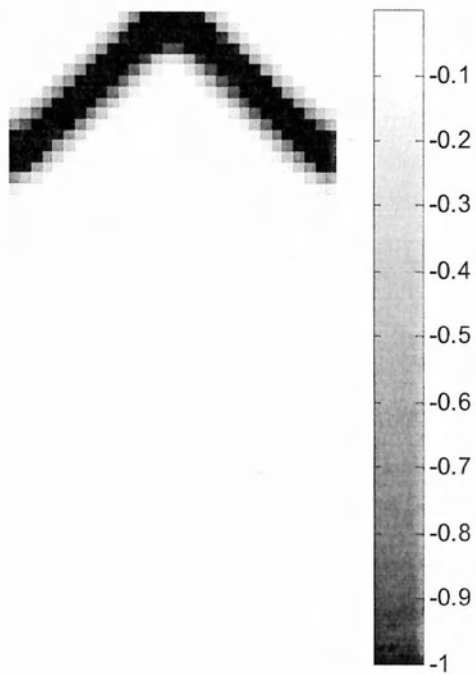


Fig. 2.2: Optimal design for volume fraction  $\text{volfrac}=0.1$ , by using the prompt line `top1(30,60,0.1,4,2.5)`

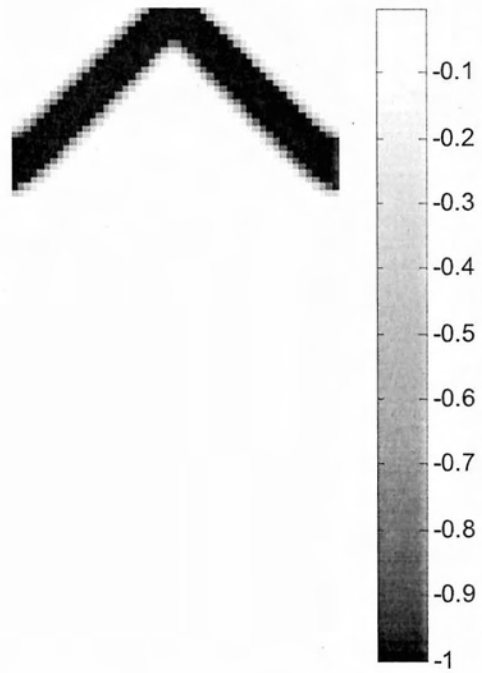


Fig.2. 3: Optimal design for volume fraction  $\text{volfrac}=0.1$ , by using the prompt line `top1(50 ,100 ,0.1 ,4 ,2.5)`



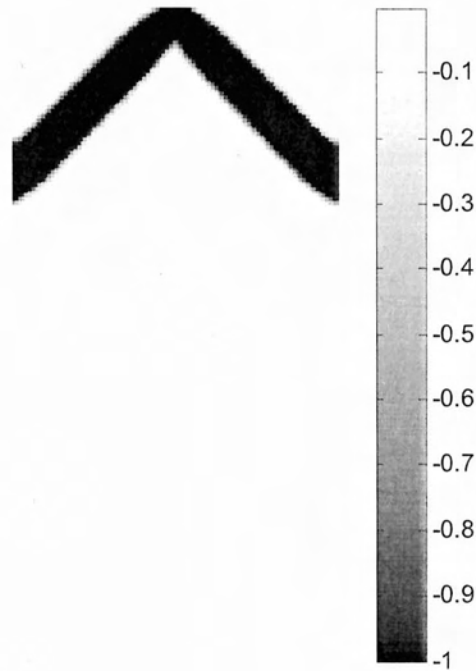


Fig. 2.4: Optimal design for volume fraction  $\text{volfrac}=0.1$ , by using the prompt line `top1(100,200,0.1,4,2.5)`

As it can be observed from the three Figures displayed above, for the same dimensions and volume ratio, the shape of the structure does not change. However, for a finer grid, which equals more elements, the design becomes clearer of grey areas. In general, it is most wanted for the design to have areas with element density closer to 0 or 1, since this way only the amount of material necessary will be used for the construction of the structure. After all, it should not be forgotten that the goal of topology optimization in general is to provide the optimum design with the least amount of material required.

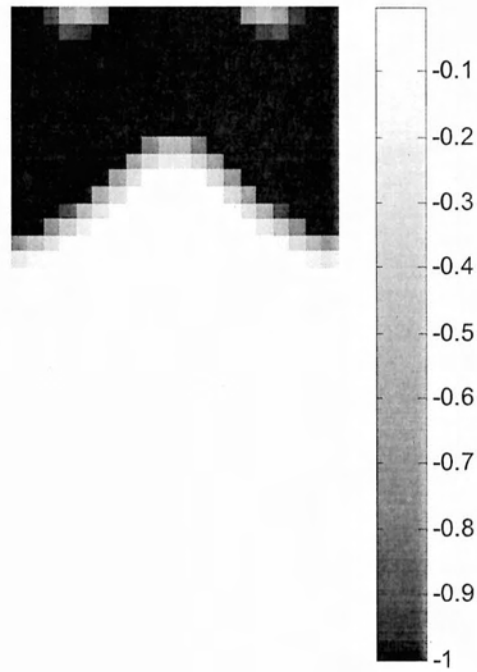


Fig.2. 5: Optimal design for volume fraction  $\text{volfrac}=0.3$ , by using the prompt line `top1(20 ,40 ,0.3 ,4 ,2.5)`

By comparing the two Figures 2.1 and 2.5 that use the same number of elements, it is obvious that, as the volume ratio ( $\text{volfrac}$ ) increases, the amount of material used increases and the design becomes more complex.

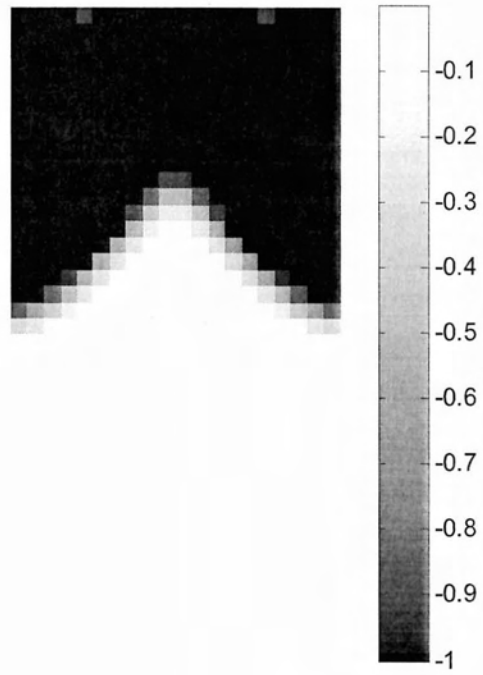


Fig.2. 6: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(20, 40, 0.4, 4, 2.5)`

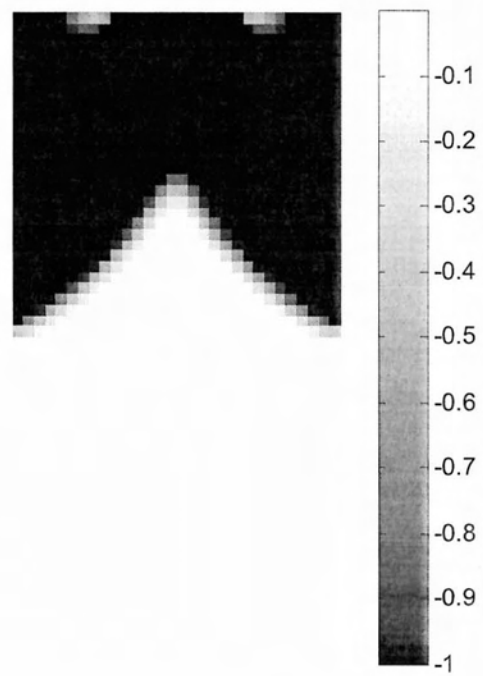


Fig. 2.7: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(30, 60, 0.4, 4, 2.5)`

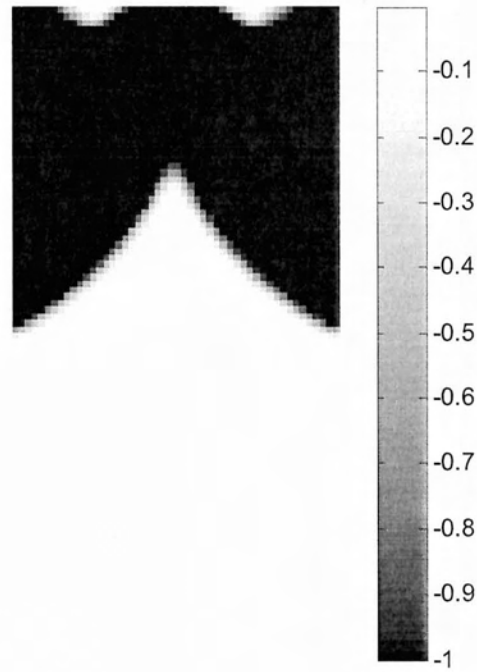


Fig. 2.8: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(50, 100, 0.4 , 4 , 2.5)`

Through Figures 2.6-2.8, for the same volume fraction, there are apparent differences. For a finer mesh, the gray regions are dramatically limited, thus leaving a smoother and more practical and applicable result. Moreover, as the number of elements increases, so does the design precision. In Figure 2.6 the holes are not clearly displayed, for a finer mesh however the design becomes optimal and the holes are much more apparent.

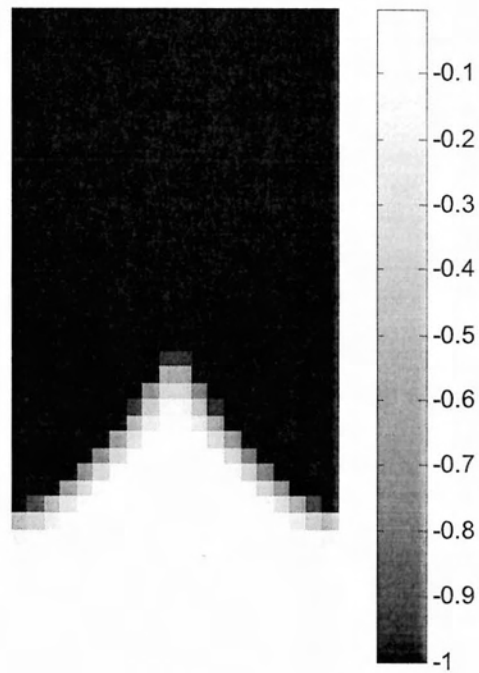


Fig. 2.9: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(20, 40, 0.7, 4, 2.5)`

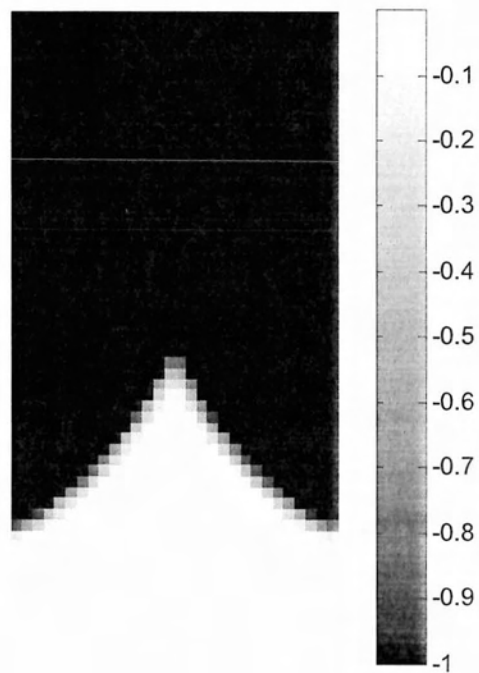


Fig. 2.10: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(30, 60, 0.7, 4, 2.5)`

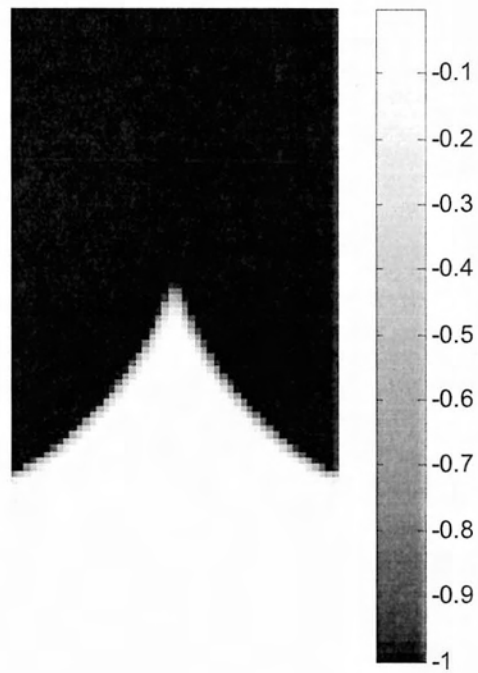


Fig. 2.11: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(50, 100, 0.7, 4, 2.5)`

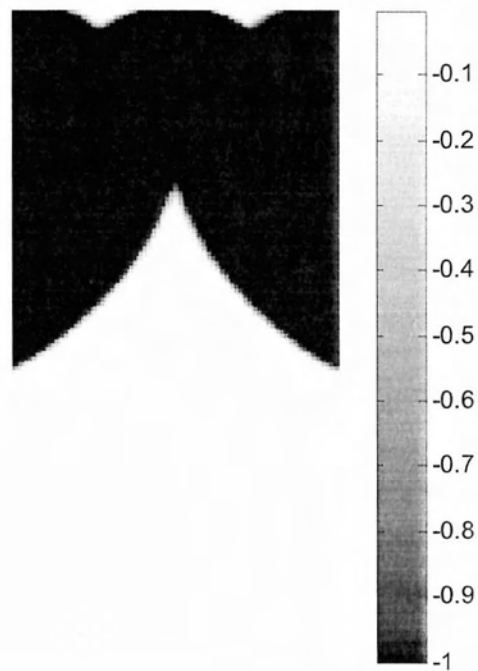


Fig. 2.12: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(100, 200, 0.7, 4, 2.5)`

The same observation that was made for Figures 2.1 and 2.5 for  $\text{volfrac}=0.1$  applies in general for all structures that have the same dimensions ratio. For a larger number of elements, the design becomes more elaborate and with minimum gray areas.

For large volume ratio  $\text{volfrac}$  equal to 0.7, from the Figures 2.9 to 2.12 the theoretically known fact that in the FEM method, for a finer mesh the results are better and more understandable is once more proved. For more elements the design variable has values 0 to 1 and avoids values around 0.5 which consume much more material than needed. This is also noticeable when comparing Figure 2.13 to 2.14 and 2.15 to 2.16 for more elements.

- For  $h / L = \text{nel}_y / \text{nel}_x = 1/2$ ,



Fig. 2.13: Optimal design for volume fraction  $\text{volfrac}=0.2$ , by using the prompt line `top1(50, 25, 0.2, 4, 2.5)`



Fig. 2.14: Optimal design for volume fraction  $\text{volfrac}=0.2$ , by using the prompt line `top1(100, 50, 0.2, 4, 2.5)`

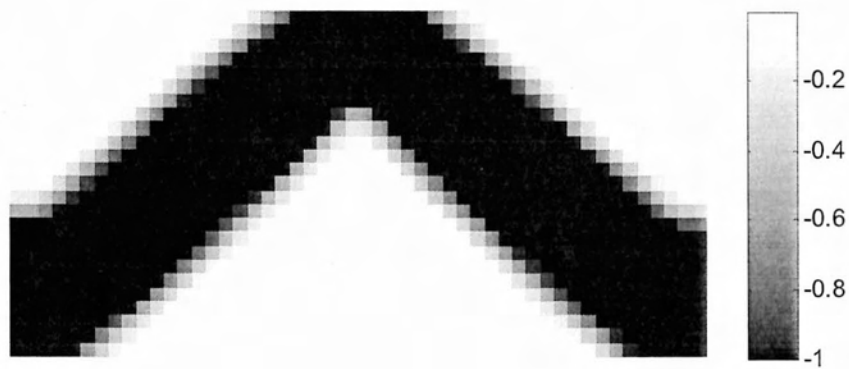


Fig. 2.15: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(50, 25, 0.5, 4, 2.5)`





Fig. 2.16: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(100, 50, 0.5, 4, 2.5)`

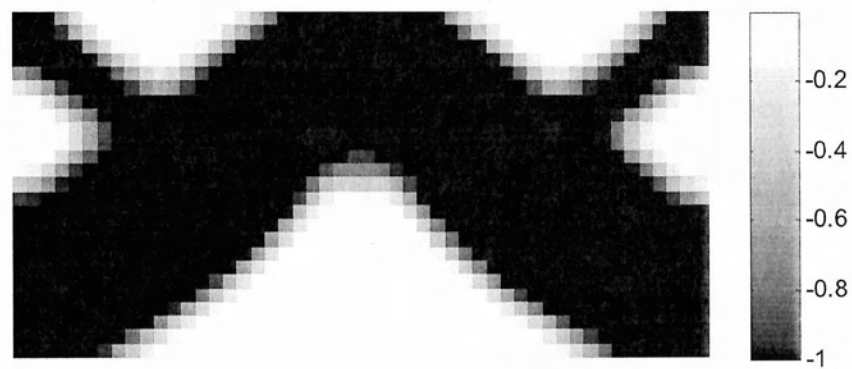


Fig.2. 17: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(50, 25, 0.7, 4, 2.5)`

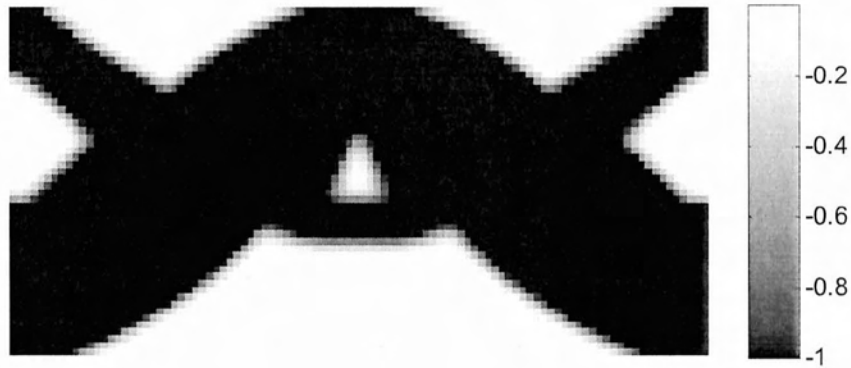


Fig. 2.18: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(100, 50, 0.7, 4, 2.5)`

By comparing Figures 2.17 and 2.18, the area around the main structure in Figure 2.17 is obviously much more blurry and incoherent than Figure 2.18. Although Figure 2.17 provides a general idea about the topology of the structure, the result is neither economical nor applicable, while Figure 2.18 provides much more necessary information on the topology.

- For  $h / L = \frac{n_{ely}}{n_{elx}} = 1/3$  ,



Fig.2. 19: Optimal design for volume fraction volfrac=0.2, by using the prompt line top1(60, 20, 0.2, 4, 2.5)



Fig. 2.20: Optimal design for volume fraction volfrac=0.2, by using the prompt line top1(120, 40, 0.2, 4, 2.5)



Fig. 2.21: Optimal design for volume fraction  $\text{volfrac}=0.2$ , by using the prompt line `top1(300, 100, 0.2, 4, 2.5)`



Fig. 2.22: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(60, 20, 0.5, 4, 2.5)`



Fig.2. 23: Optimal design for volume fraction volfrac=0.5, by using the prompt line top1(120, 40, 0.5, 4, 2.5)



Fig.2. 24: Optimal design for volume fraction volfrac=0.5, by using the prompt line top1(300, 100, 0.5, 4, 2.5)



Fig. 2.25: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(60, 20, 0.7, 4, 2.5)`



Fig. 2.26: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(120, 40, 0.7, 4, 2.5)`

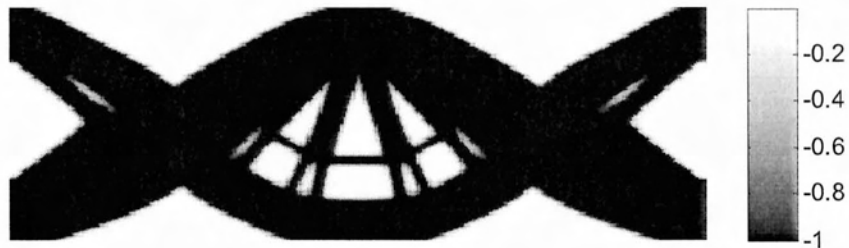


Fig.2. 27: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(300, 100, 0.7, 4, 2.5)`

Figures 2.22 to 2.24 and 2.25 to 2.27 are a characteristic example of detailed topology optimization design. Although Figures 2.23 and 2.26 seem in their respective examples to be optimal designs, it is proved that as the total number of elements increases and the mesh becomes more detailed, even more holes appear to produce a finer and more sophisticated result.

2. Fixed left end with load applied in the upper right corner of the beam:

- For  $h/L = nely/nelx=2$ ,

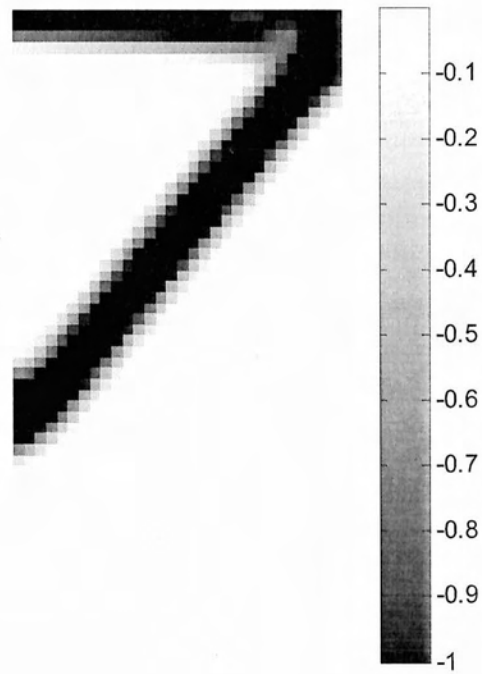


Fig. 2.28: Optimal design for volume fraction volfrac=0.2, by using the prompt line top1(30, 60, 0.2, 4, 2.5).



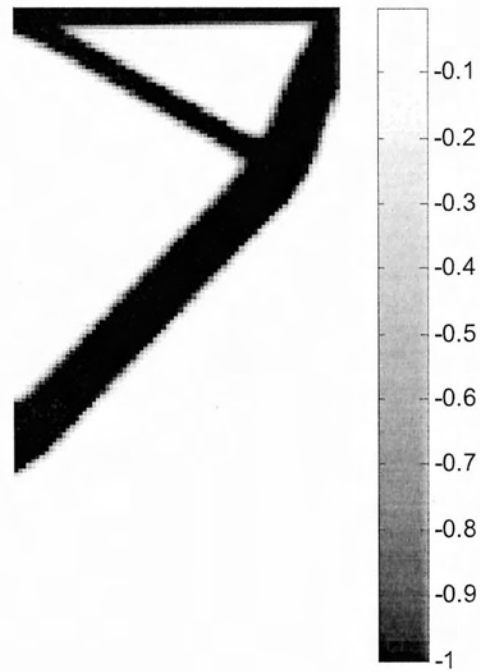


Fig. 2.29: Optimal design for volume fraction  $\text{volfrac}=0.2$ , by using the prompt line `top1(75, 150, 0.2, 4, 2.5)`.

The evolution of the design in Figure 2.28 is obvious. At the upper side of the structure, there are gray areas which for more elements evolve into a clear beam with 100% material element density.

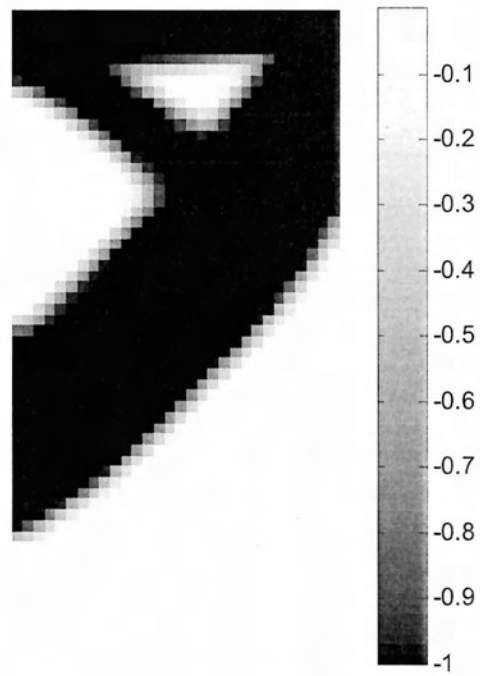


Fig. 2.30: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(30, 60, 0.5, 4, 2.5)`.

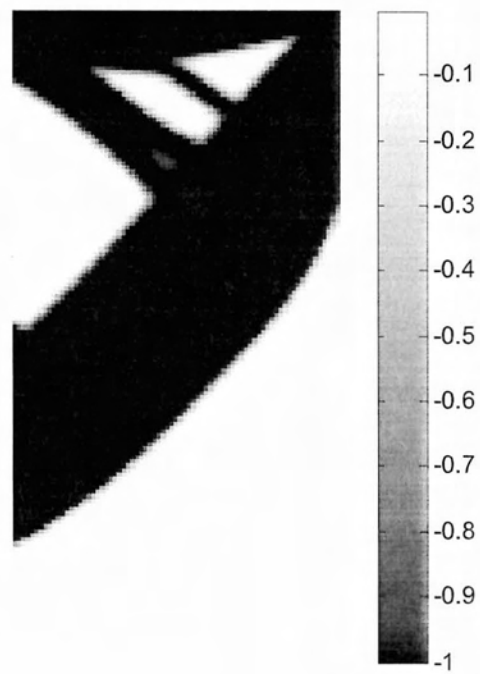


Fig. 2.31: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(75, 150, 0.5, 4, 2.5)`.

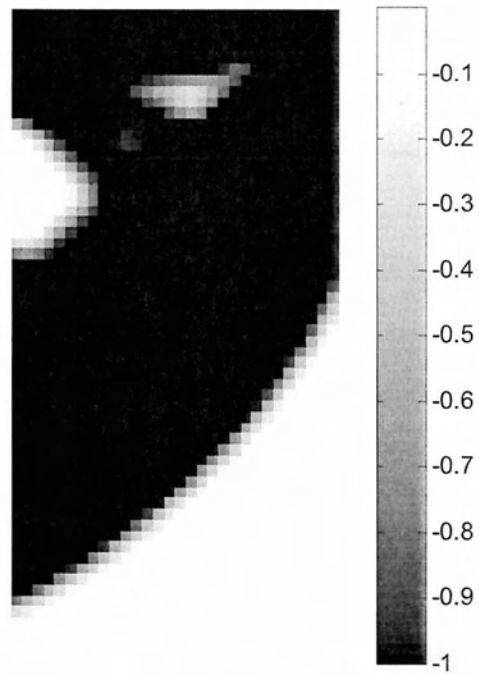


Fig.2. 32: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(30, 60, 0.7, 4, 2.5)`.

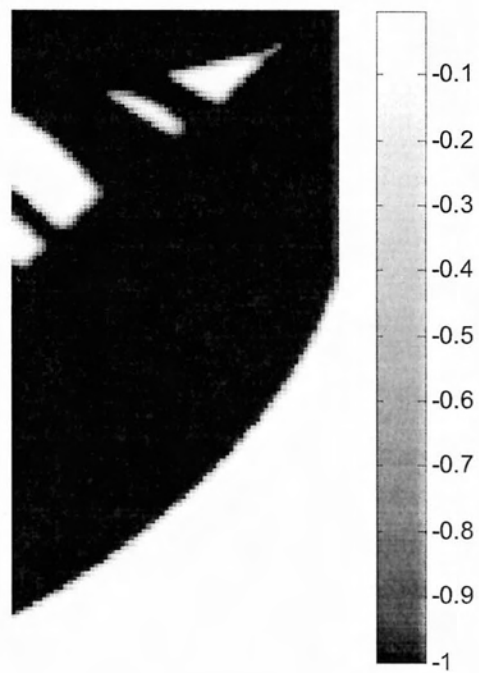


Fig. 2.33: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(75, 150, 0.7, 4, 2.5)`.

By comparing Figures 2.30 to 2.31 and 2.32 to 2.33 the gray areas for this boundary conditions have evolved to holes which make the design more detailed and restrain the use of material.

- For  $h / L = \frac{nely}{nelx} = 5$ ,

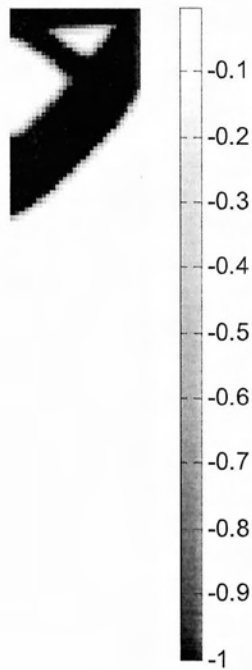


Fig. 2.34: Optimal design for volume fraction volfrac=0.2, by using the prompt line top1(30, 150, 0.2, 4, 2.5).

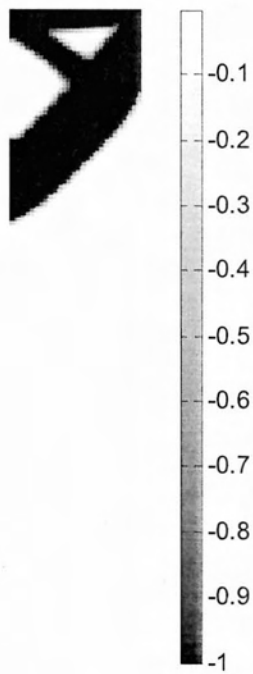


Fig. 2.35: Optimal design for volume fraction  $\text{volfrac}=0.2$ , by using the prompt line `top1(40, 200, 0.2, 4, 2.5)`.

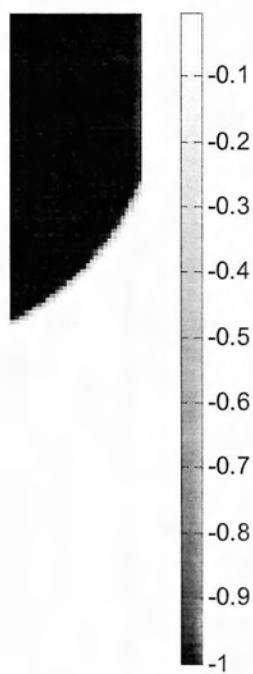


Fig. 2.36: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(30, 150, 0.4, 4, 2.5)`.

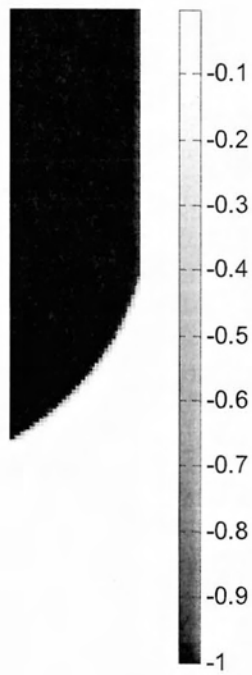


Fig. 2.37: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(40, 200, 0.4, 4, 2.5)`.

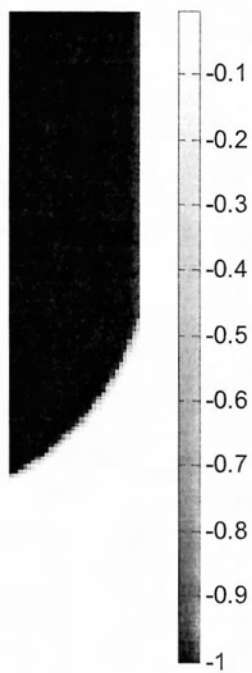


Fig. 2.38: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(30, 150, 0.7, 4, 2.5)`.

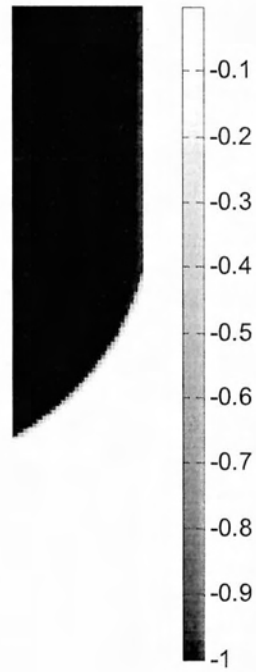


Fig. 2.39: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(40, 200, 0.7, 4, 2.5)`.

- For  $h / L = \text{nely} / \text{nelx} = 1/2$ ,

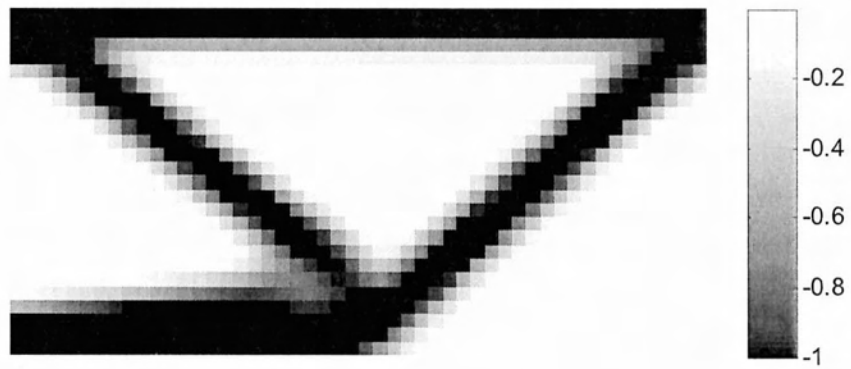


Fig. 2.40: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(50, 25, 0.4, 4, 2.5)`.



Fig. 2.41: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(150, 75, 0.4, 4, 2.5)`.



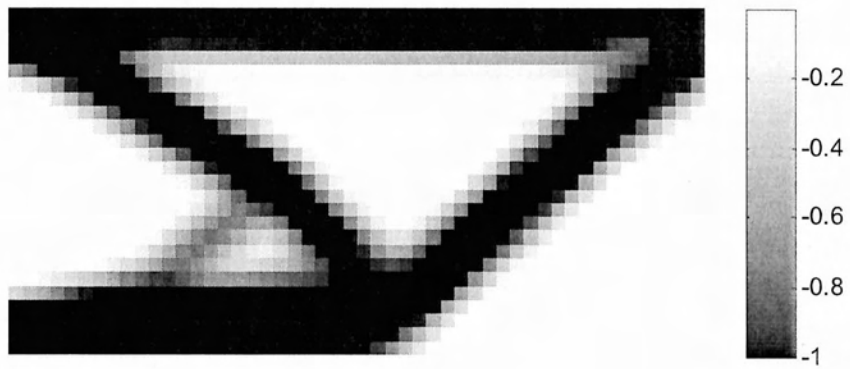


Fig. 2.42: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(50, 25, 0.5, 4, 2.5)`

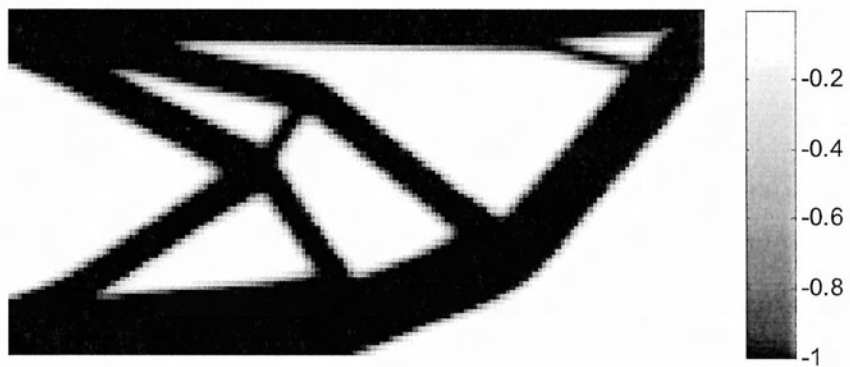


Fig. 2. 43: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(150, 75, 0.5, 4, 2.5)`

Figure 2.40 is more incoherent than Figure 2.41 and through the iterations of the program it validates the observation that gray areas of material improve to either 0 or 1 and provide a realizable construction.

3. **Fixed left end with load applied uniformly on the upper side of the beam:**

- For  $h / L = n_{ely} / n_{elx} = 2$ ,

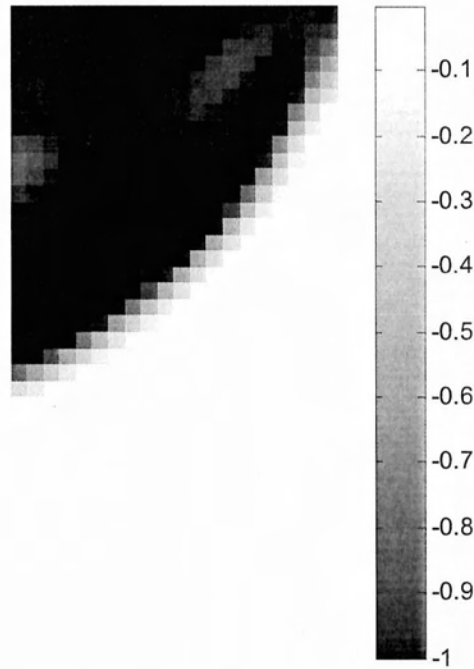


Fig. 2.44: Optimal design for volume fraction volfrac=0.4, by using the prompt line top1(20, 40, 0.4, 4, 2.5)

In Figure 2.44 the appearance of holes is seen on pixel level, which is not practical from an engineering side, since it demands more material and thus increases the cost. In Figures 2.45 and 2.46 the material element density has better distribution.

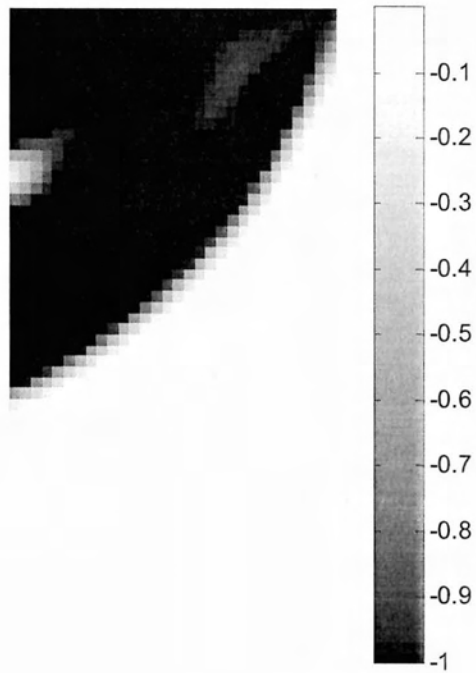


Fig. 2.45: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(30, 60, 0.4, 4, 2.5)`

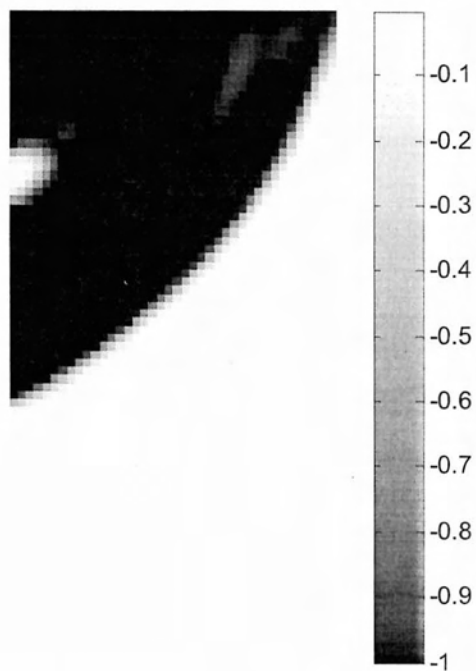


Fig. 2.46: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(40, 80, 0.4, 4, 2.5)`

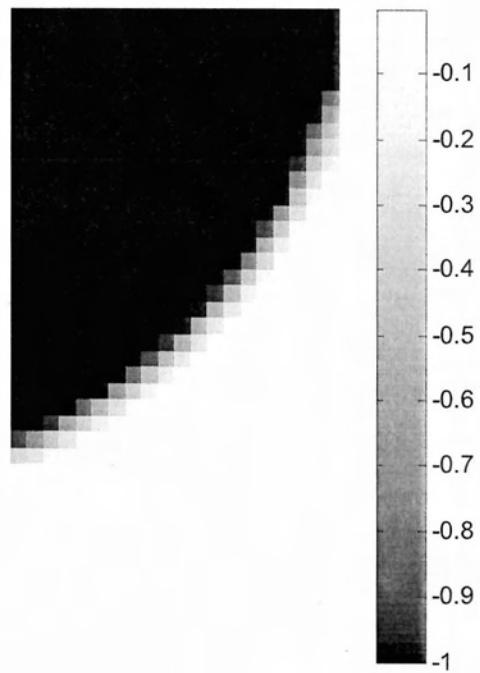


Fig. 2.47: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(20, 40, 0.5, 4, 2.5)`

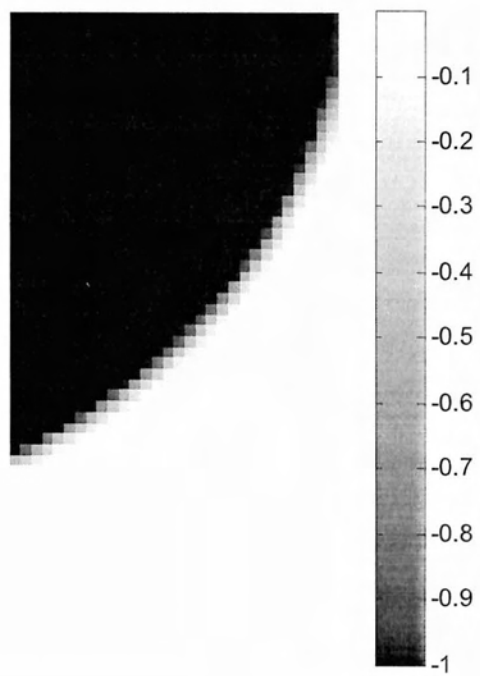


Fig. 2.48: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(30, 60, 0.5, 4, 2.5)`

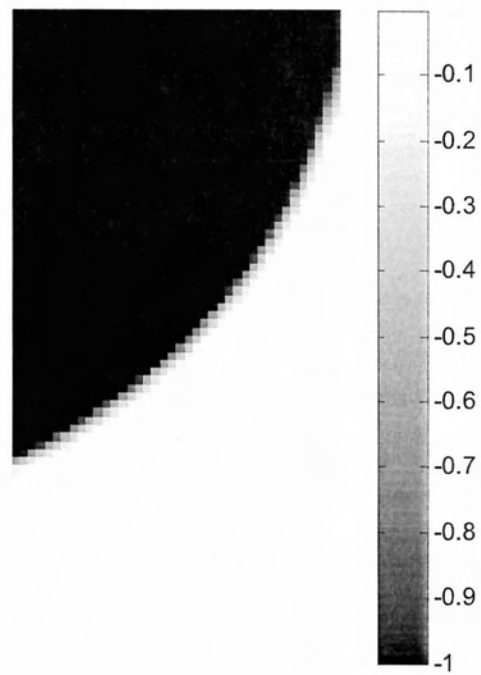


Fig.2. 49: Optimal design for volume fraction volfrac=0.5, by using the prompt line top1(40, 80, 0.5, 4, 2.5)

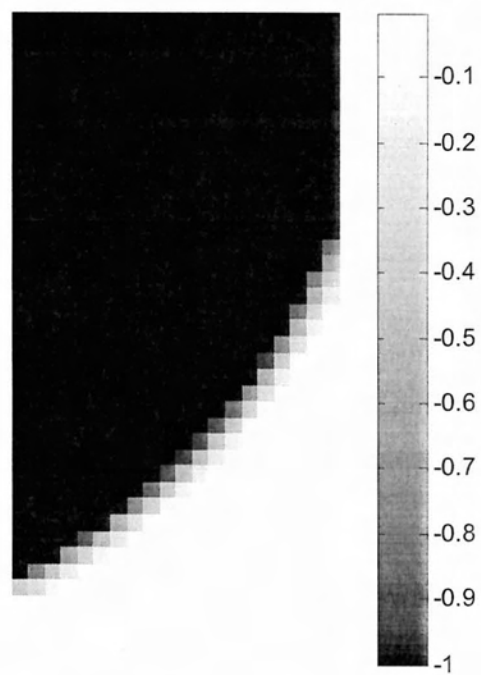


Fig. 2.50: Optimal design for volume fraction volfrac=0.7, by using the prompt line top1(20, 40, 0.7, 4, 2.5)

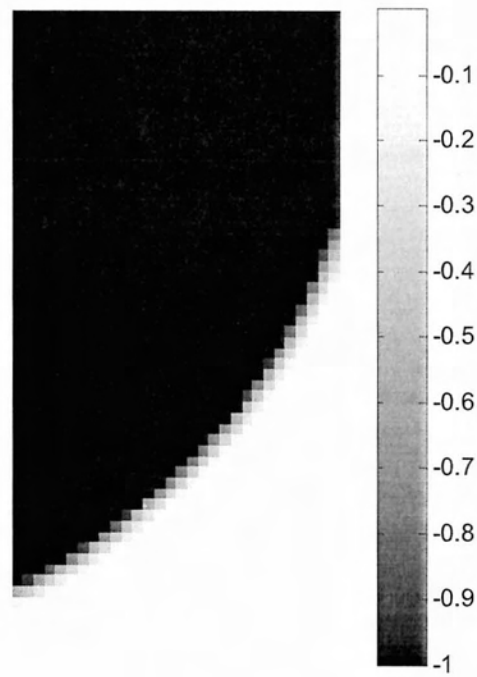


Fig. 2.51: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line  $\text{top1}(30, 60, 0.7, 4, 2.5)$

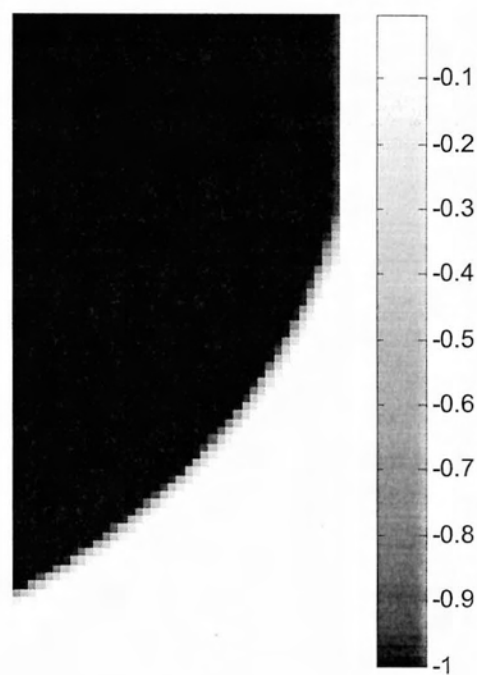


Fig.2. 52: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line  $\text{top1}(40, 80, 0.7, 4, 2.5)$

4. Scrolling on the lower left side and joint on the lower right side of the beam:

- For  $h / L = nely / nelx = 5$ ,

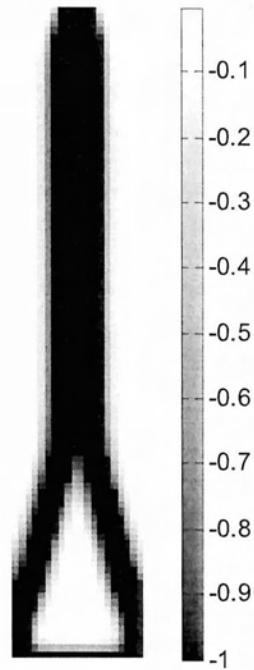


Fig. 2.53: Optimal design for volume fraction volfrac=0.5, by using the prompt line top1(20, 100, 0.5, 4, 2.5)

For this boundary condition, as also for the previous, the Matlab code **top1** distributes the material according to the direction of the load. In Figures 2.53 to 2.67, the structure is symmetrical to the load applied in the middle and the structure is mostly laid out in the opposite direction of the load, on the node where the load is applied. It makes perfect sense for structures with such boundary conditions to have this form, because for less material the resistance of the structure to the strength remains the same.

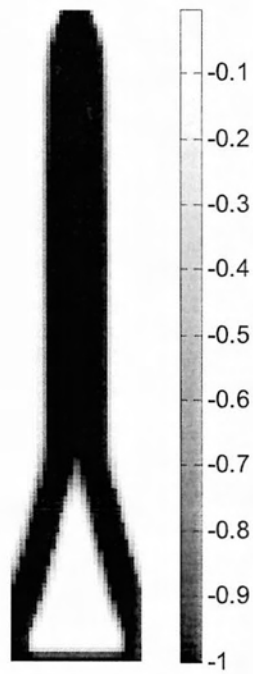


Fig. 2.54: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(30, 150, 0.5, 4, 2.5)`

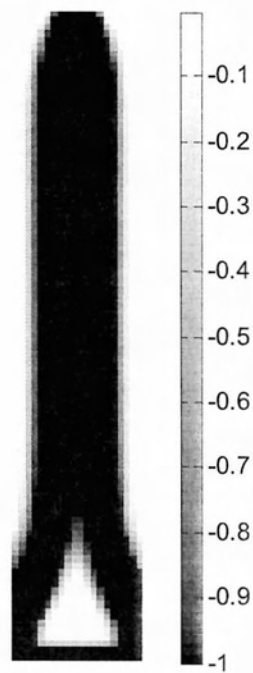


Fig. 2.55: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(20, 100, 0.7, 4, 2.5)`



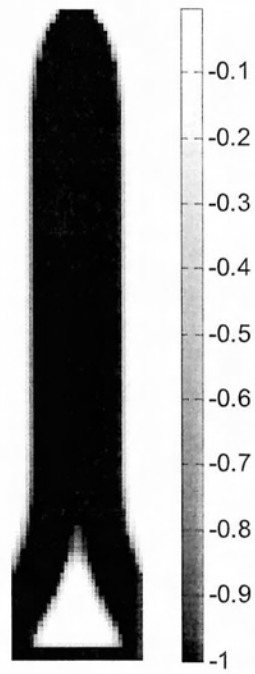


Fig. 2.56: Optimal design for volume fraction volfrac=0.7, by using the prompt line top1(30, 150, 0.7, 4, 2.5)



- For  $h / L = \frac{nely}{nelx} = 2$ ,

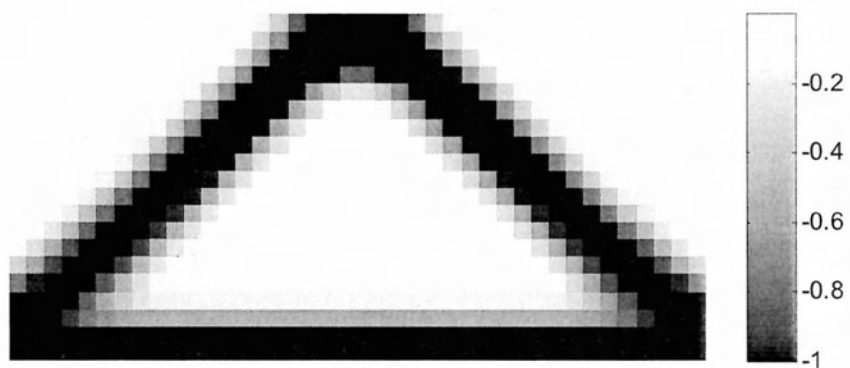


Fig. 2.57: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(40, 20, 0.4, 4, 2.5)`



Fig. 2.58: Optimal design for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(100, 50, 0.4, 4, 2.5)`

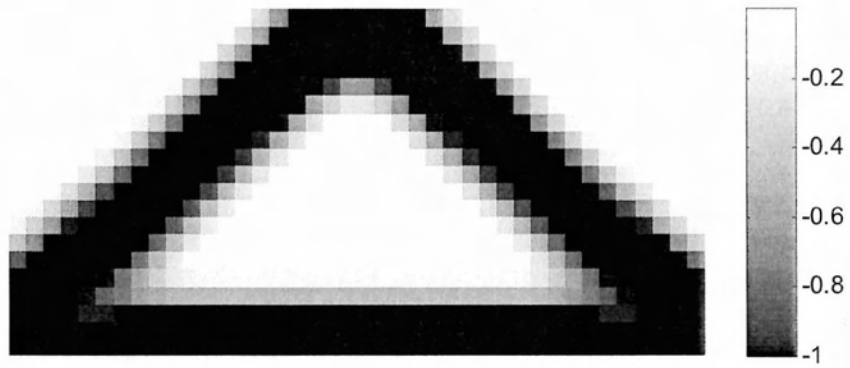


Fig. 2.59: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(40, 20, 0.5, 4, 2.5)`

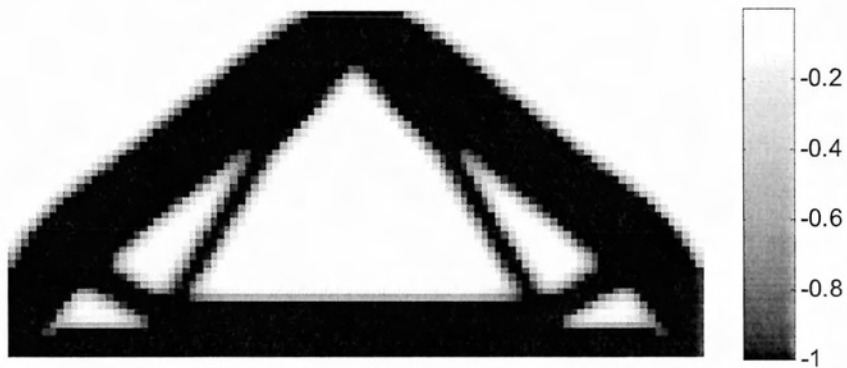


Fig. 2.60: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(100, 50, 0.5, 4, 2.5)`

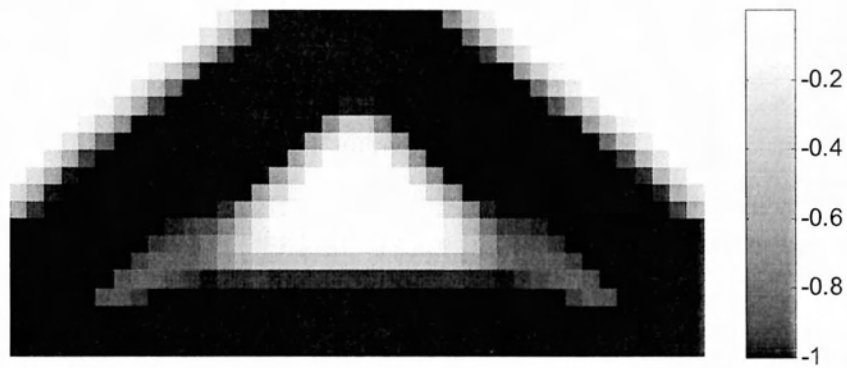


Fig. 2.61: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(40, 20, 0.7, 4, 2.5)`



Fig. 2.62: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(100, 50, 0.7, 4, 2.5)`

- For  $h / L = n_{ely} / n_{elx} = 1/2$ ,

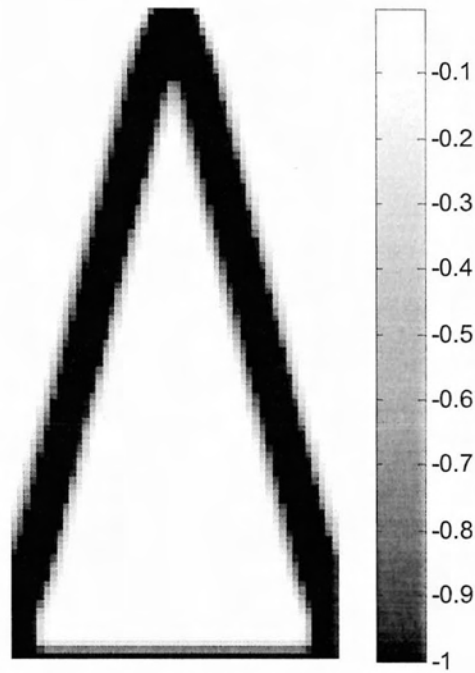


Fig. 2.63: Optimal design for volume fraction volfrac=0.3, by using the prompt line top1(50, 100, 0.3, 4, 2.5)

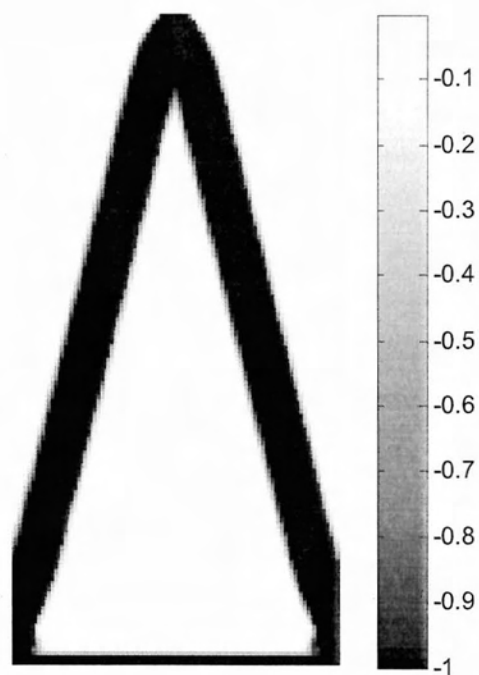


Fig. 2.64: Optimal design for volume fraction  $\text{volfrac}=0.3$ , by using the prompt line `top1(100, 200, 0.3, 4, 2.5)`

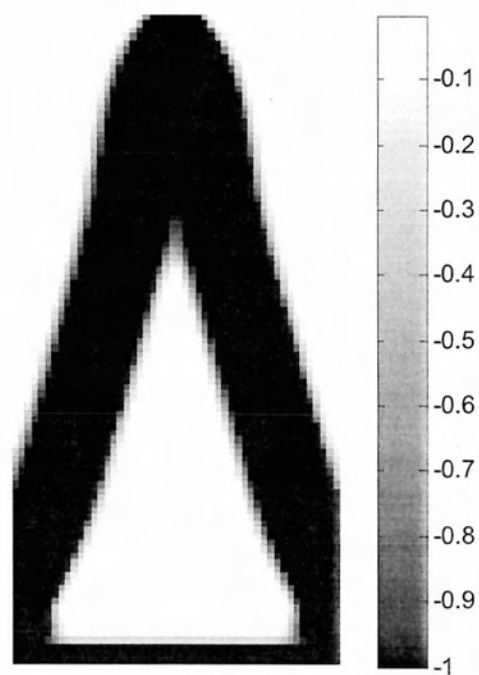


Fig. 2.65: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(50, 100, 0.5, 4, 2.5)`

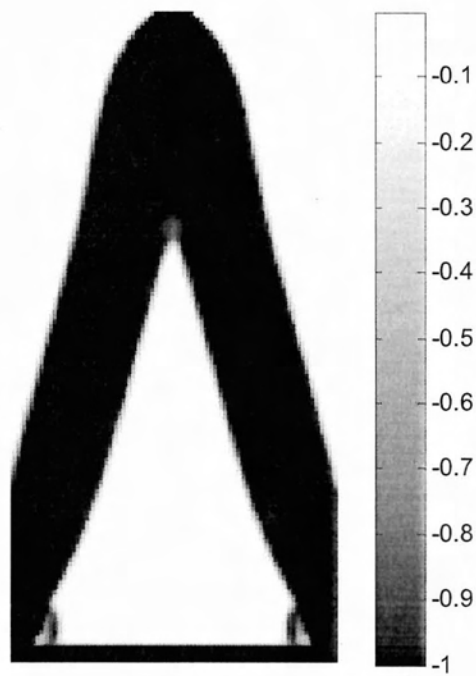


Fig. 2.66: Optimal design for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(100, 200, 0.5, 4, 2.5)`

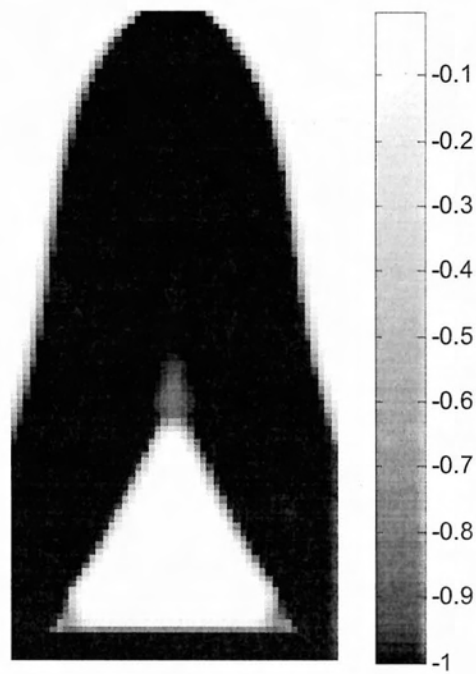


Fig. 2.67: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(50, 100, 0.7, 4, 2.5)`

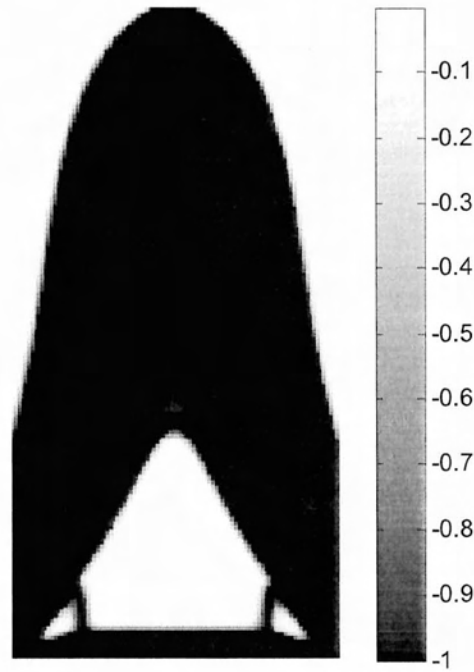


Fig. 2.68: Optimal design for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(100, 200, 0.7, 4, 2.5)`

By studying the Figures shown above, the following observations can be made:

- ✓ As the number of elements increases, so does the quality of the shape, in terms of grey areas. Fewer grey areas mean less material to be used, and therefore, less expensive structure.
- ✓ Apart from the number of elements, it is shown that more complex designs (with holes, for instance) appear for a larger number of volume ratio.
- ✓ Another observation that needs to be made concerns the influence of the variables **penal** and **rmin** to the quality of the design. In the Figures shown above, the values of these two variables remained constant, because the parameters under study were the dimensions and volume ratio. Some examples out of many that shows the difference for an increase in **penal** and **rmin** are the following:



From Figures 2.69 and 2.70, it can be seen that by increasing **penal** and **rmin**, the design seems more 'blurry' and less accurate, due to the appearance of more grey regions. The hole that appeared in Figure 2.69 does not exist in Figure 2.70 that has a larger number of **rmin**. In Figures 2.75 and 2.76, it can also be noticed that there are more elements with density lower than 0.8 around the main optimal structure for a higher value of **rmin**. Those cause the 'blurry' effect. This effect is found to be mainly attributed to the increase of the value of **rmin** to 3.5. From the realization of more examples, the ideal value of **rmin** was found to be around 2.5, but no more than 3. As Sigmund mentioned in his work (1999), 'by selecting **rmin** less than one, the filtered sensitivities will be equal to the original sensitivities, making the filter inactive' (citation from Sigmund, 1999).

Moreover, by observing Figures 2.71 and 2.72, where only the **penal** value was altered, there is an apparent difference. For **penal** equal to 4.5, the structure shows more holes at the lower part of the structure, thus making it more complex. The clarity of the structure that concerns the grey areas with less density has, however, remained intact. The ideal value for **penal** was found between 3.5 to 4. The increase in **penal** up to 4 improves the quality of the design concerning the grey areas. However, for a value larger than 4, the results do not show any further improvement. Instead, as it can be observed from Figures 2.73 and 2.74, there are no problems with grey areas and with the exterior shape of the structure. However, for a large value at **penal**, like 7, the global stiffness matrix is close to becoming singular or badly scaled and inside the structure there are many differences. Therefore, the results may prove inaccurate.



Fig. 2.69: Optimal design (fixed at both ends) for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(60, 20, 0.7, 4, 2.5)`



Fig. 2.70: Optimal design (fixed at both ends) for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(60, 20, 0.7, 4.5, 3.5)`



Fig. 2.71: Optimal design (scrolling and joint) for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(100, 50, 0.7, 4, 2.5)`



Fig. 2.72: Optimal design (scrolling and joint) for volume fraction  $\text{volfrac}=0.7$ , by using the prompt line `top1(100, 50, 0.7, 4.5, 2.5)`



Fig. 2.73: Optimal design (fixed left end) for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(150, 75, 0.5, 4, 2.5)`



Fig. 2.74: Optimal design (fixed left end) for volume fraction  $\text{volfrac}=0.5$ , by using the prompt line `top1(150, 75, 0.5, 7, 2.5)`.

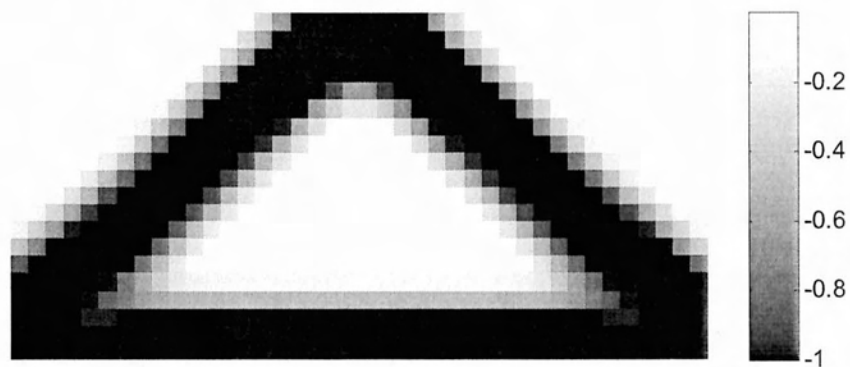


Fig.2. 75: Optimal design (scrolling and joint) for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(40, 20, 0.4, 4, 2.5)`

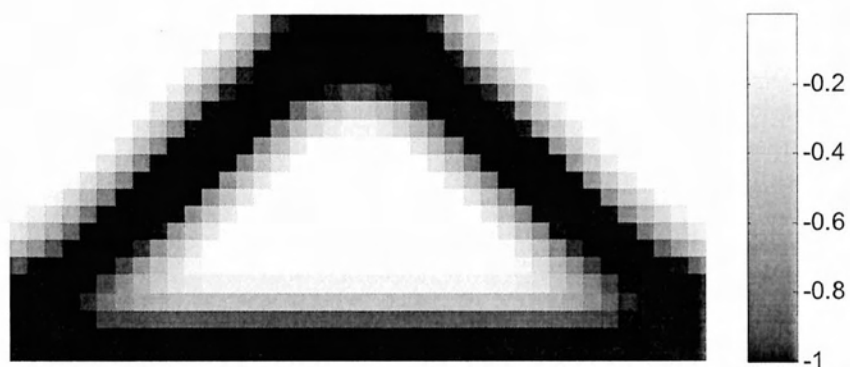


Fig. 2.76: Optimal design (scrolling and joint) for volume fraction  $\text{volfrac}=0.4$ , by using the prompt line `top1(40, 20, 0.4, 4.5, 3.5)`



# Chapter 3

## The robust load case

### 3.1 Theoretical background for robust topology optimization

In Chapter 2, the topology optimization problem for the deterministic load case was studied for different boundary conditions. In addition, the results for different values of the variables used to call the program were examined. In this Chapter, the topology problem is examined after uncertainty in load has been inserted, according to the theoretical background by the paper of Papadimitriou and Papadimitriou (2015). This comes as a necessary evolution of the deterministic load case, since structures are, in reality, sensitive to loading uncertainties. Loading uncertainties arise from the lack of knowledge in the magnitude and direction of concentrated loads and also in the spatial variability of loads applied on the structure (Papadimitriou and Papadimitriou, 2015).

The topology optimization for the robust approach is now formulated for objective functions related to uncertainty measures of the compliance of the structure. These uncertainty measures include mean value of the load (for this case of robust load), standard deviation and level exceedance (Papadimitriou and Papadimitriou, 2015). This approximation requires the calculation of multi-dimensional integrals over the uncertain parameter space, which are evaluated using the following methods:

- Sparse grid quadrature techniques for the mean and standard deviation,
- The approximate first-order reliability method (FORM) for the failure probability.

Alternatively, the multi-dimensional integrals are estimated by using Taylor series expansion techniques and Monte Carlo methods. However, by using Taylor series expansion, the accuracy might be lost for bigger uncertainties. Monte Carlo (MC) methods, although they increase the computational cost, secure the independence of the

number of uncertain parameters. So the MC methods solve the problem for large number of uncertain parameters, without increasing the computational cost like Taylor expansion and Gauss quadrature techniques do. Grid methods have also been applied to robust design, since they are quite accurate, model non-intrusive and easily parallelized. The integrals are, as mentioned above, estimated using Gauss quadrature methods, which transform the robust topology optimization problem into a weighted sum of deterministic problems at the Gauss quadrature points (Smolyak, 1963; Zenger, 1997; Gerstner et al., 1998; Griebel et al., 2004). This brings up the need for the estimation of the sensitivities of the objective function using the adjoint method at these Gauss quadrature points. Nevertheless, however accurate the grid methodologies may be, they increase greatly the computational cost in case of a great number of uncertain parameters. This is why the sparse-grid approach is used, which keeps the number of functions evaluations low, without harming the accuracy of the obtained results.

In robust topology optimization, the objective functions to be minimized are formulated as the weighted sum of the expected performance and the standard deviation of the performance. The constraints refer to the volume of the structure (like the deterministic model) and the state equations. Since the topology optimization problem requires a very large number of design variables, the use of adjoint methods is required to compute the sensitivities with respect to the uncertain parameters (Jameson, 1988).

As mentioned above, the robust topology optimization problem is formulated as the minimization of the compliance with constraint on the maximum allowed material, and it is presented below (Papadimitriou and Papadimitriou, 2015).

$$\begin{aligned}
\min_{\rho} \quad & F(\rho, \mathbf{v}) = \int_{\Omega} f_i(\mathbf{v}) u_i(\rho, \mathbf{v}) d\Omega + \int_{S_1} g_i(\mathbf{v}) u_i(\rho, \mathbf{v}) dS \\
s.t. \quad & C = \int_{\Omega} \rho d\Omega \leq V_f \\
and \quad & \frac{\partial \sigma_{ij}(\rho, \mathbf{v})}{\partial x_j} + b_i(\boldsymbol{\theta}) = 0 \quad , \mathbf{x} \in \Omega \\
& \sigma_{ij}(\rho, \mathbf{v}) n_j = t_i(\boldsymbol{\theta}) \quad , \mathbf{x} \in S_1 \\
& u_i(\rho, \mathbf{v}) = 0 \quad , \mathbf{x} \in S_2
\end{aligned} \tag{3.1}$$

where  $\rho \equiv \rho(x)$  are the local densities and also the design variables,  $f_i(v)$  and  $g_i(v)$  are general functions of the uncertainty function  $v \equiv v_k, k=1, \dots, m$  ( $m$  is the number of Gaussian quadrature points) defined on the domain  $\Omega$ .  $S = S_1 \cup S_2$  represents the surface of the body, as known from Mechanics of materials theory,  $u_i(v, \rho)$  and  $\sigma_{ij}(v, \rho)$  are the displacement and stress components, respectively. Finally,  $V_f$  is a part of the total volume.

The problem expressed above in (3.1) includes the uncertainty for more than one cases and also takes in consideration the stresses. In this work, only the loading uncertainty is taken into consideration, so only parts of (3.1) will be needed. First, the mean and standard deviation of the problem have to be expressed containing the design variable  $\rho$  and the load uncertainty  $v \equiv v_k, k=1, \dots, m$ . The mean value is given in terms of the first two statistical moments of the multidimensional performance function  $F$ .

$$\mu_{m,F}(\rho) = \int_V [F(\rho, \mathbf{v})]^m p(\mathbf{v}) d\mathbf{v} \quad (3.2)$$

$$\mu_F = \mu_{1,F} \quad (3.3)$$

$$\sigma_F = \sqrt{\mu_{2,F} - \mu_F^2} \quad (3.4)$$

$$m = 1, 2.$$

Following the methodology from the work of Papadimitriou and Papadimitriou, (2015), a good approximation for the integrals shown above is given by the Gauss-Hermite quadrature on sparse grids. By assuming that the uncertain parameters follow a Gaussian distribution, the (3.2) integral becomes:

$$\int_V F(\rho, \mathbf{v}) p(\mathbf{v}) d\mathbf{v} \simeq \sum_{k=1}^n w_k F(\rho, \mathbf{v}_k) \quad (3.5)$$

where  $v_k$  is the uncertainty at the  $k$  sparse grid point and  $w_k$  are the weighting coefficients. The robust objective function is:

$$G(\mathbf{F}) = F_{rob} = w \mu_{1,F}(\mathbf{F}) + (1 - w) \sqrt{\mu_{2,F}(\mathbf{F}) - \mu_{1,F}^2(\mathbf{F})} \quad (3.6)$$

Following the logic of the deterministic problem, the first order sensitivities to the design parameters  $\rho(x)$  is:

$$\frac{\partial G(\mathbf{F})}{\partial \rho} = \sum_{k=1}^n \frac{\partial G(\mathbf{F})}{\partial F_k} \frac{\partial F_k}{\partial \rho} = \sum_{k=1}^n a_k \frac{\partial F_k}{\partial \rho} \quad (3.7)$$

where

$$a_k = \frac{\partial G(\mathbf{F})}{\partial F_k} = w w_k + (1 - w) w_k \frac{F_k - \mu_{1,F}(\mathbf{F})}{\sigma_F(\mathbf{F})} \quad (3.8)$$

are the coefficients and  $F_k$  are the values of the performance function  $F$  at the grid points (Papadimitriou and Papadimitriou, 2015).. Another way that provides the same result is to find the derivative of the standard deviation of the problem

$$\frac{\partial \sigma}{\partial \rho} = \frac{\frac{\partial \mu_{2,F}}{\partial \rho} - 2\mu_{1,F} \frac{\partial \mu_{1,F}}{\partial \rho}}{2\sqrt{(\mu_{2,F} - \mu_{1,F}^2)}} \quad (3.9)$$

as well as

$$\frac{\partial \mu_{2,F}}{\partial \rho} = \sum_{k=1}^n 2w_k F_k \frac{\partial F_k}{\partial \rho} \quad (3.10)$$

$$\frac{\partial \mu_{1,F}}{\partial \rho} = \sum_{k=1}^n w_k F_k^2 \quad (3.11)$$

and finally

$$\frac{\partial F_{Rob}}{\partial \rho} = w \frac{\partial \mu_{1,F}}{\partial \rho} + (1 - w) \frac{\partial \sigma}{\partial \rho} \quad (3.12)$$

### 3.2 Implementation in Matlab

The methodology explained above is implemented in a topology optimization code and will be explained briefly in this Chapter. This code is built in Matlab according again to the one for topology optimization by Sigmund (1999) and is called by the prompt line:

**top2(nelx, nely, volfrac, penal, rmin, w, vmean, vsigma)**

for the case that one load is applied to the structure and:

**run\_top2n(nelx, nely, volfrac, penal, rmin, w, vmean1, vmean2, vsigma1, vsigma2)**

for the two-load case. As far as the variables are concerned, **vmean** is the mean value of the load, **vsigma** is the standard deviation of the load and **w** is a weighting factor determined by the user, according to the desired influence of the standard deviation. The reason that different programs are needed for the estimation of the uncertainty on the structure is that there are different mean load values and for the same standard deviation different **vsigma1** and **vsigma2**.

The robust topology optimization program begins by loading from text files the number of sparse grid points **m**, the weighting factors for each sparse grid point **wd** and the sparse grid point locations **xd**. The sparse grid point information are obtained by programs that can be found online for sparse grid Hermite. Matlab loads the file set by the user, according to the accuracy required in calculations (for example w4.dat) and saves the data into column matrices **wd** and **xd**, respectively (**wd** and **xd1**, **xd2** for the two load case). Then, the topology optimization procedure is followed with the same logic as in the static case problem, with the difference that an additional **for** loop is added within **while** loop. This must be done so that the program can estimate on each optimization iteration the loading uncertainties for each sparse grid point. The uncertainty is implemented into the code through the [FE] subroutine, where the boundary conditions are determined.

1. For the one-load case, the load boundary condition for load at the upper right corner is:

$$F(2*(nely+1)*(nelx+1)-2*nely,1)=vmean+sqrt(2)*vsigma*xd(n);$$

2. For the two-load case, the load boundary condition for load at the upper right corner and for another load in the lower side of the structure and in the middle is:

$$F(2*(nely+1)*(nelx+1)-2*nely,1)=vmean1+sqrt(2)*vsigma1*xd1(n);$$

and

$$F((nely+1)*(nelx+2),1)=vmean2+sqrt(2)*vsigma2*xd2(n);$$

respectively. The subroutine [FE] is called **m** more times on each iteration, which increases the computational cost and makes the computational time much bigger. On each iteration, the mean values that will be needed in order to calculate the mean value and the standard deviation of the problem are calculated for each grid point. The mean values are also multiplied by a constant  $1/\pi^{m/2}$ , where **m** in this equation depends on the number of loads applied. So for the one-load case, **m=1** and  $1/\sqrt{\pi}$  and for the two-load case, **m=2** and  $1/\pi$ . After that, the standard deviation and the derivatives **dsigma** and **dmean1** are estimated by the equations (3.9), (3.10) and (3.11). From equations (3.6) and (3.12) the robust objective function **Frob** and its first order derivative **dFrob** are, respectively, calculated. The program continues by calling the function **check** for the filtering of sensitivities of **dFrob**. Finally, the function **OC** is called to update the design via the optimality criteria method and the results are printed, like the deterministic problem.

### 3.3 Results and discussion

In Chapter 3.1, the methodology for estimating the optimal topology for the insertion of uncertainty in load was presented. In Chapter 3.2, this methodology was implemented into the existing topology Matlab code by Sigmund (1999) and two programs were studied; one for the case of the application of a single load and another for the application of two loads. These two programs were ran for various boundary conditions and values of the variable  $w$  and the following results were retrieved:

1. Fixed left end with load applied in the upper right corner of the beam:

- **Deterministic case:**

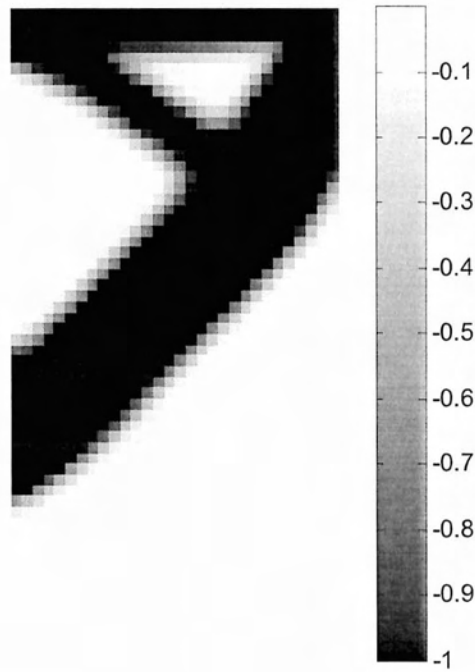


Fig. 3.1: Optimal design for the static case problem, by using the prompt line `top1(30, 60, 0.4, 4, 2.5)`.

- **Uncertain case:** For  $m=7$  sparse grid points (x2.dat)

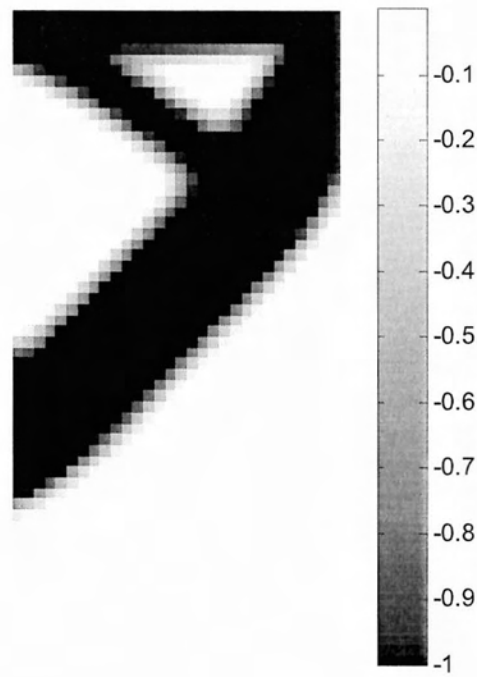


Fig. 3.2: Optimal design for the uncertain case problem, by using the prompt line `top2(30, 60, 0.4, 4, 2.5, 0.5, -1, 0.3)`

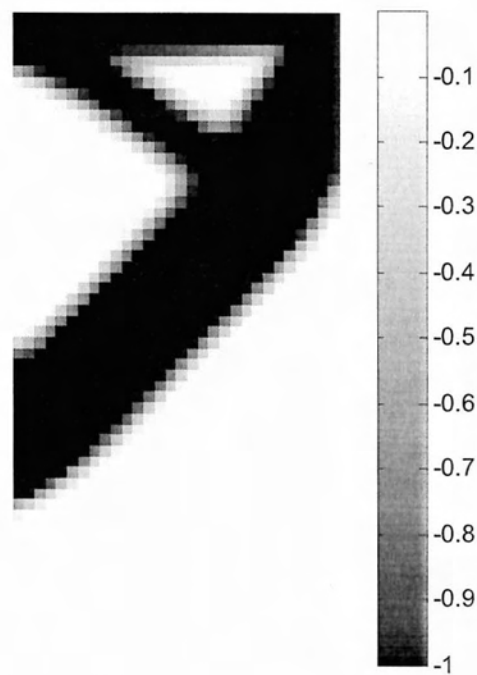


Fig.3.3: Optimal design for the uncertain case problem, by using the prompt line `top2(30, 60, 0.4, 4, 2.5, 0.6, -1, 0.2)`



From Figures 3.1 and 3.2 and 3.3, it can be observed that, although the problems are those of a static case and robust, thus different, the topology of the structure does not change in general. This observation, although at first can be startling, has a very simple explanation; despite the change in load, there is only one load applied and the topology does not change for the given structure. In this example, there is no symmetry as far as the boundary conditions are concerned. This can be verified from more examples with different boundary conditions.

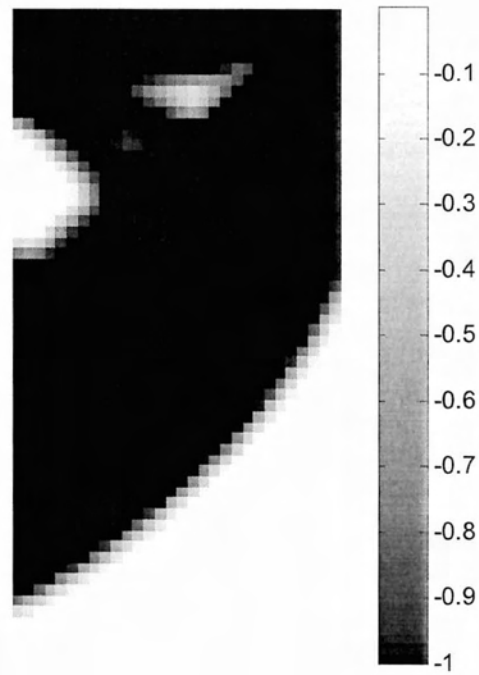


Fig.3.4: Optimal design for the static case problem, by using the prompt line `top1(30, 60, 0.7, 4, 2.5)`.

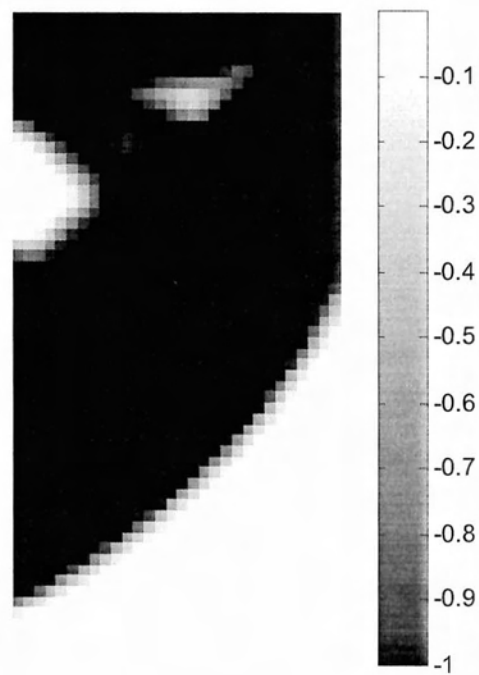


Fig.3.5: Optimal design for the uncertainty problem, by using the prompt line `top2(30, 60, 0.7, 4, 2.5, 0.8,-1,0.2)`

In Figures 3.4 and 3.5 a slight difference can be observed on pixel level in the holes of the structure. As mentioned before, this type of problem is asymmetrical, however not strongly enough to present a big difference so the structure does not present any topology alterations. Since for a slightly asymmetrical example the uncertainty code does not provide any topological difference, it is understood that for a symmetrical case like the fixed ends at both sides and load in the middle problem there will be no apparent difference.

2. Fixed left end with load applied in the upper right corner of the beam:



Fig.3.6: Optimal design for the static case problem, by using the prompt line `top1(100, 50, 0.5, 4, 2.5)`



Fig. 3.7: Optimal design for the uncertainty problem, by using the prompt line `top2(100,50,0.5,4, 2.5, 0.3,-1,0.35)`

As expected, for this symmetrical problem the topology remains exactly the same. This comparison between symmetrical and asymmetrical problems as well as the fact that, although the uncertainty was inserted in the problem no apparent differences were produced, brought out the need for a problem with more than loads to be studied. However, in the case of two loads, each load has different sparse grid point locations. Moreover, for different values of each load, the standard deviation for each load changes. The **top2** code was modified for that cause and so the **run\_top2n** code was created for two loads. Figure 3.8 demonstrates the deterministic example with two loads, one at the upper right corner and the other at the lower side in the middle. Figures 3.9 and 3.10 demonstrate the uncertainty examples with different standard deviation each.

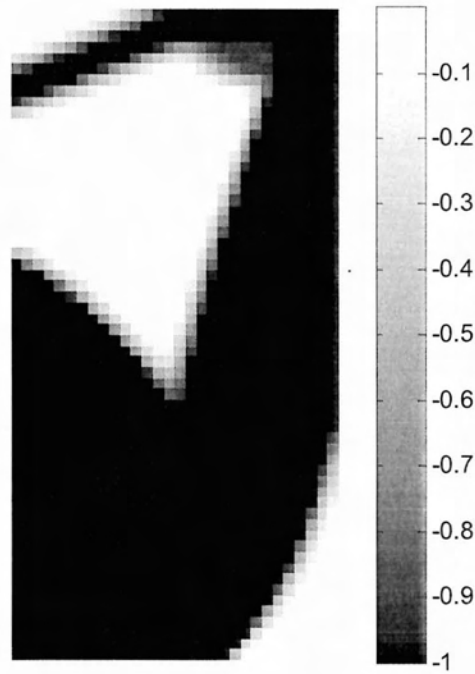


Fig. 3.8: Optimal design for the static case problem, by using the prompt line `top1(30, 60, 0.7, 4, 2.5)`

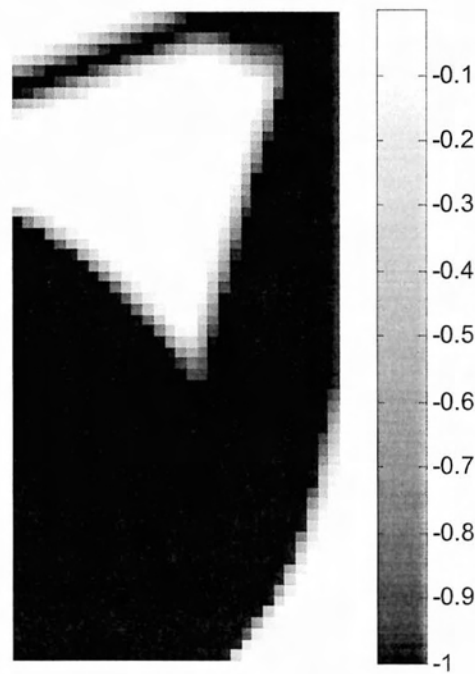
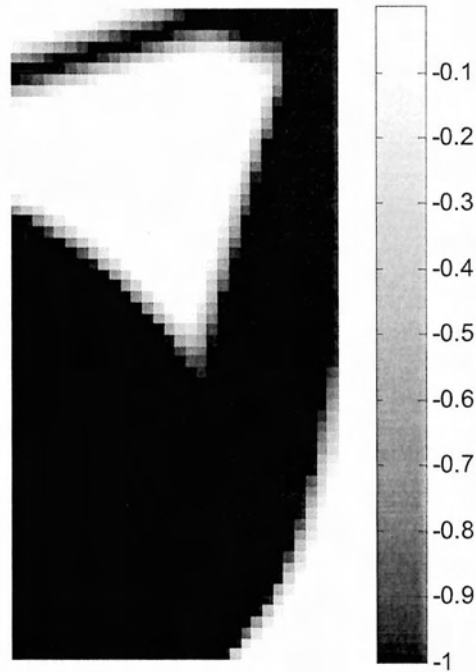


Fig.3.9: Optimal design for the uncertainty problem, by using the prompt line `run_top2n(30,60,0.7,4,2.5, 0.3,-1,3,0.3,0.9)`



**Fig. 3.10: Optimal design for the uncertainty problem, by using the prompt line `run_top2n(30,60,0.7,4,2.5, 0.3,-1,3,0.35,1.05)`**

From Figures 3.8, 3.9 and 3.10, the difference is apparent when the uncertainty is inserted. As the standard deviation of each load is increasing, we can observe parts of the structure getting slimmer. However, since the volume fraction (volfrac) has a steady value, there is no loss of material. The material that has left the slimmer areas appears to have moved to the bulkier bottom side of the structure. This can be attributed in this example, to the values of the loads, since the bottom load value is three times larger, so more material is needed there.

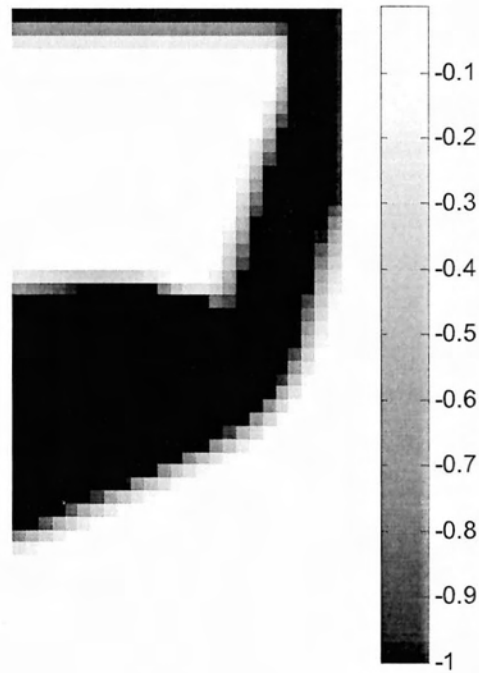


Fig. 3.11: Optimal design for the static case problem, by using the prompt line `top1(25, 50, 0.4, 4, 2.5)`

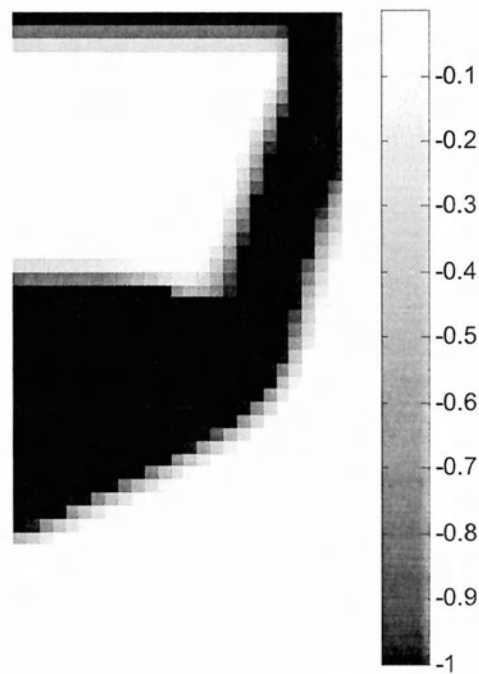
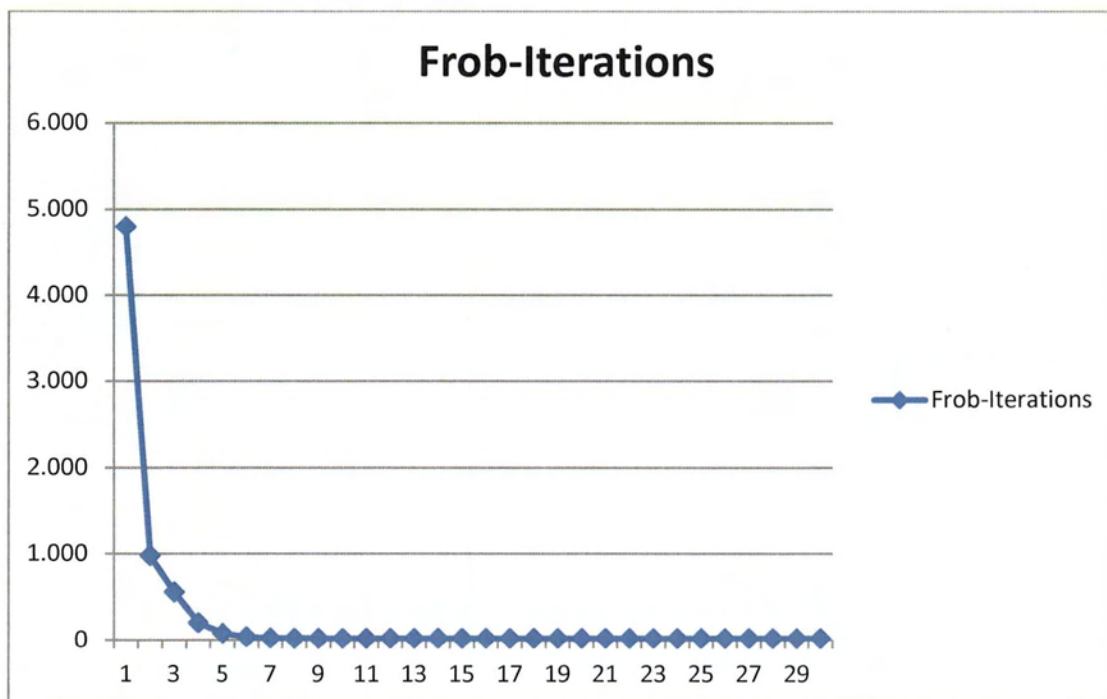


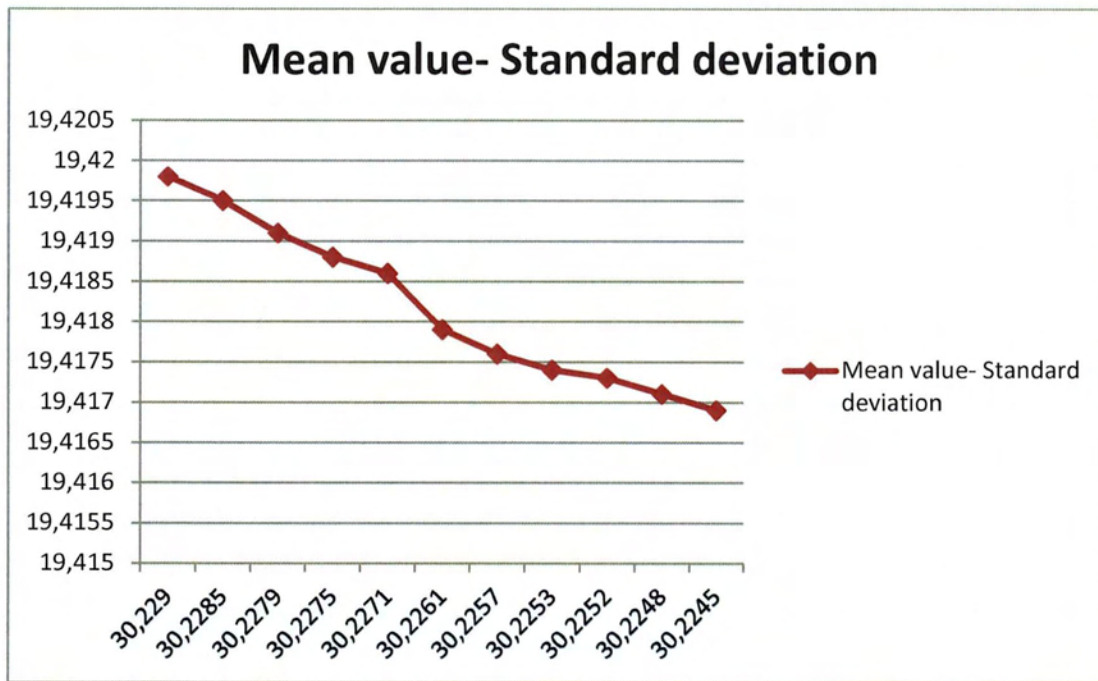
Fig.3.12: Optimal design for the uncertainty problem, by using the prompt line `run_top2n(25,50,0.4,4,2.5, 0.3,-1,3,0.35,1.05)`

Between the Figures 3.11 and 3.12 there are also differences at the interior of the structure and more specifically at its lower point. For the static load case, there appears to be more material at the direction of the force. At the uncertain case, the material is not concentrated in the middle where the load is applied, like the static case. Due to the load uncertainty, the material seems more even and less concentrated to one direction.

From the results produced above, the case with fixed end at the left side and load at the upper right corner was studied further. For a 30x60 mesh and various values for the weighting factor of the objective function  $w$ , the convergence history of the compliance objective function is shown at the graph below. It is observed that in the beginning, the objective function has a very high value because the design variable is equal to the volume fraction. As the iterations continue, the value of the objective function decreases significantly and finally converges to a much smaller value.







For different values of the weighting factor  $w$ , a Pareto front was drawn for the optimal mean value and the standard deviation. From the Pareto it can be noticed that as the mean value decreases, the standard deviation increases. This is attributed to the equation (3.4). When the mean value becomes smaller, the standard deviation in (3.4) becomes larger and vice-versa. However, the range of variation in the values of the mean and standard deviation is small.

# Chapter 4

## Concluding remarks

This thesis examined the topology optimization of structures under deterministic and uncertain load. For its realization, a code was used that was developed in Matlab by Sigmund (1999). Alterations were made in the code in order to compare the original structure with the optimal given by the code. Three versions of the 99 line code by Sigmund were developed. The first one, **top1** was tested for different load and boundary conditions, as well as for finer mesh. Different values for the variable **penal** and **rmin** were also examined. Finally, using the theory by Mechanics of materials, a stress and strain calculation algorithm was implemented into the original code.

From the examples performed, the following observations were made:

- For a finer mesh, the optimal structure becomes more complex and with less gray areas, meaning a better distribution of material.
- By increasing the **penal** variable above the value 4.5 the gray areas are not affected. However, if increased too much, the topology may change and prove inaccurate. The ideal value for **penal** is around 4.
- The ideal value for **rmin** is 2.5. For a larger value, there appear more gray areas.

It is worth noticing that the observations made for the deterministic model also apply to the robust model, since the role of the variables **penal** and **rmin** does not change.

Regarding the uncertainty, two codes were built for the topology optimization. Code **top2** is built for the case of one-load uncertainty and code **run\_top2n** is built for uncertainty in two loads. For the uncertainty in one load only, the topology does not change in comparison to the static case problem, especially for the symmetrical problem for fixed ends at both sides and load in the middle. For more apparent alterations, a more asymmetrical model was required and so the two load problem was inserted. The results

were indeed more obvious and the alterations between the deterministic and the robust problem were mainly observed at the point where the forces were applied.

Although this thesis reaches its aims, a few recommendations for future work can be made, based on the findings of this project:

- Topology optimization for more than two loads and for very asymmetrical cases.
- Implementation of material uncertainty in the topology optimization framework.

## BIBLIOGRAPHY

1. S. Smolyak. (1963). 'Quadrature and interpolation formulas for tensor products of certain classes of functions', *Doklady Akademii Nauk SSSR*, 4; 240-243.
2. M.P. Bendsoe, N. Kikuchi. (1988). Generating optimal topologies in structural design using a homogenization method, *Computer Methods in Applied Mechanics and Engineering*. 71; 197-224, Elsevier.
3. A. Jameson. (1988). Aerodynamic design via control theory, *Journal of scientific Computing*, 3(3), 233-260.
4. M.P. Bendsoe. (1989). Shape optimization as a discrete optimization procedure- Solution procedures using continuous approximations, *Discretization Methods and Structural Optimization- Procedures and Applications*. 42; 40-47, Springer.
5. M. Zhou, G.I.N. Rozvany. (1991). The COC algorithm, part II: topological, geometrical and generalized shape optimization. *Computer Methods in Applied Engineering and Mechanics*. 89; 309-336, Elsevier.
6. H.P. Mlejnek. (1992). Some aspects of the genesis of structures. *Structural and Multidisciplinary Optimization*. 5; 64-69, Springer.
7. C. Zenger. (1997). Sparse grids. In Proceedings of the Research Workshop of the Israel Science Foundation on Multiscale Phenomenon, *Modelling and Computation*, (p. 86).
8. T. Gerstner, M. Griebel. (1998). 'Numerical integration using sparse grids', *Numerical Algorithms*, 18;209-232, Springer.
9. J. Albery, C. Carstensen, S. Funken. (1999). Remarks around 50 lines of Matlab: short Finite Element implementation, *Numerical Algorithms*. 20(2-3); 117-137, Springer.
10. M.P. Bendsoe, O. Sigmund. (1999). Material interpolations in Topology Optimization, *Archive of Applied Mechanics*, 69; 635-654, Springer.
11. O. Sigmund, (2001). A 99 line topology optimization code written in Matlab, *Structural and Multidisciplinary Optimization*, 21; 120-127, Springer.
12. M.P. Bendsoe, O. Sigmund, (2003). *Topology Optimization: Theory, Methods and Applications*, 2nd edition, Springer.
13. H.J. Bungartz, M. Griebel, (2004). 'Sparse Grids', *Acta numerica*, 13; 147-269.

14. H.S. Jung, S. Cho, (2004). Reliability-based topology optimization of geometrically nonlinear structures with loading and material uncertainties', *Finite Elements in Analysis and Design*, 41(3); 311-331, Elsevier.
15. V.J. Challis. (2010). A discrete level-set Topology Optimization code written in Matlab, *Structural and Multidisciplinary Optimization*. 41(3); 453-464, Springer.
16. K. Perros (2010). Design Optimization of Structures subjected to robust excitations, Doctoral Thesis, University of Thessaly.
17. E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, O. Sigmund, (2011). Efficient topology optimization in Matlab using 88 lines of code, *Structural and Multidisciplinary Optimization*, 43; 1-16, Springer.
18. M. Jalalpour, K. J. Guest, T. Igusa, (2013). Reliability-based topology optimization of trusses with robust stiffness, *Structural Safety*, 43; 41-49, Elsevier.
19. J.C. Medina, A. Taflanidis, (2014). Probabilistic measures for assessing appropriateness of robust design optimization solutions, *Structural and Multidisciplinary Optimization*, 1-22, Springer.
20. L. Xia. P. Breitkopf. (2014). A Reduced Multiscale Model for Nonlinear Structural Topology Optimization, *Computer Methods in Applied Engineering and Mechanics*, 280; 117-134, Elsevier.
21. X. Huang, Y. Li, S.W. Zhou, Y.M. Xie. (2014). Topology Optimization of Compliant Mechanisms with desired structural stiffness, *Engineering Structures*, 79; 13-21, Elsevier.
22. W. Zhang, W. Zhong, X. Guo. (2014). An explicit length scale control approach in SIMP-based topology optimization, *Computer Methods in Applied Engineering and Mechanics*, 282; 71-86, Elsevier.
23. A.N. Christiansen, J.A. Baerentzen, M. Nobel-Jorgensen, N. Aage, O. Sigmund. (2015). Combined Shape and Topology Optimization of 3D Structures, *Computers and Graphics*, 46; 25-35, Elsevier.
24. M. Jansen, G. Lombaert, M. Schevenels, (2015). Robust topology optimization of structures with imperfect geometry based on geometric nonlinear analysis, *Computer Methods in Applied Engineering and Mechanics*, 285; 452-467, Elsevier.
25. V. Shobeiri. (2015). The Topology Optimization Design for cracked structures, *Engineering Analysis with Boundary Elements*, 58; 26-38, Elsevier.

26. L. Xia, P. Breitkopf. (2015). Multiscale Structural Topology Optimization with an approximate constitutive model for local material, *Computer Methods in Applied Engineering and Mechanics*, 286; 147-167, Elsevier.
27. Y. Wang, Z. Luo, J. Wu, N. Zhang. (2015). Topology Optimization of Compliant Mechanisms using Element-Free Galerkin Method, *Advances in Engineering Software*, 85; 61-72, Elsevier.
28. M. Carrasco, B. Ivorra, A.M. Ramos. (2015). Robust Topology Design Optimization for continuous elastic materials, *Computer Methods in Applied Engineering and Mechanics*, 289; 131-154, Elsevier.
29. C. Papadimitriou, D. Papadimitriou. (2015). Robust and Reliability-based structural topology optimization using a continuous adjoint method, *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, (submitted).

## APPENDIX

### Program for the static load case

```

function topl(nelx,nely,volfrac,penal,rmin)
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
% START ITERATION
while change > 0.01
    loop = loop + 1;
    xold = x;
% FE-ANALYSIS
    [U]=FE(nelx,nely,x,penal);
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    [KE] = lk;
    c = 0.;
    for ely = 1:nely
        for elx = 1:nelx
            n1 = (nely+1)*(elx-1)+ely;
            n2 = (nely+1)* elx +ely;
            Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2;
2*n1+1;2*n1+2],1);
            c = c + x(ely,elx)^penal*Ue'*KE*Ue;
            dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
        end
    end
% FILTERING OF SENSITIVITIES
    [dc] = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
    [x] = OC(nelx,nely,x,volfrac,dc);
% PRINT RESULTS
    change = max(max(abs(x-xold)));
    disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c)
...
        ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
        ' ch.: ' sprintf('%6.3f',change )])
% PLOT DENSITIES
    colormap(gray); imagesc(-x); axis equal; axis tight; axis
off;pause(1e-6);
end
%%STRESS AND STRAIN CALCULATION%%%%
Sy=1.0;
nu=0.3;
E = 1;
Sxx=0; Sxy=0; Syy=0;
    exx=0; exy=0; eyy=0;
for ely=1:nely
    for elx=1:nelx

        if x(ely,elx)>=0.500
            n1=(nely+1)*(elx-1)+ely;
            n2=(nely+1)*elx+ely;
            %%Displacement for Horizontal degrees of freedom

```

```

        %Ut=(1/4)*(1-q)*(1+n)*U(2*n1-
1,:) +(1/4)*(1+q)*(1+n)*U(2*n2-1,:) +(1/4)*(1+q)*(1-
n)*U(2*n2+1,:) +(1/4)*(1-q)*(1-n)*U(2*n1+1,:);
        %%Displacement for Vertical degrees of freedom
        %Vt=(1/4)*(1-
q)*(1+n)*U(2*n1,:) +(1/4)*(1+q)*(1+n)*U(2*n2,:) +(1/4)*(1+q)*(1-
n)*U(2*n2+2,:) +(1/4)*(1-q)*(1-n)*U(2*n1+2,:);
        %exx=-(1/4)*(1+n)*U(2*n1-1,:) +(1/4)*(1+n)*U(2*n2-
1,:) +(1/4)*(1-n)*U(2*n2+1,:) -(1/4)*(1-n)*U(2*n1+1,:);
        %eyy=(1/4)*(1-q)*U(2*n1,:) +(1/4)*(1+q)*U(2*n2,:) -
(1/4)*(1+q)*U(2*n2+2,:) -(1/4)*(1-q)*U(2*n1+2,:);
        %exy=(1/2)*((1/4)*(1-q)*U(2*n1-1,:) +(1/4)*(1+q)*U(2*n2-
1,:) -(1/4)*(1+q)*U(2*n2+1,:) -(1/4)*(1-q)*U(2*n1+1,:) -
(1/4)*(1+n)*U(2*n1,:) +(1/4)*(1+n)*U(2*n2,:) +(1/4)*(1-n)*U(2*n2+2,:) -
(1/4)*(1-n)*U(2*n1+2,:));
        exx=-(1/4)*(2)*U(2*n1-1,:) +(1/4)*(2)*U(2*n2-
1,:) +(1/4)*(2)*U(2*n2+1,:) -(1/4)*(2)*U(2*n1+1,:);
        eyy=(1/4)*(2)*U(2*n1,:) +(1/4)*(2)*U(2*n2,:) -
(1/4)*(2)*U(2*n2+2,:) -(1/4)*(2)*U(2*n1+2,:);
        exy=(1/2)*((1/4)*(2)*U(2*n1-1,:) +(1/4)*(2)*U(2*n2-1,:) -
(1/4)*(2)*U(2*n2+1,:) -(1/4)*(2)*U(2*n1+1,:) -
(1/4)*(2)*U(2*n1,:) +(1/4)*(2)*U(2*n2,:) +(1/4)*(2)*U(2*n2+2,:) -
(1/4)*(2)*U(2*n1+2,:));
        Sxx=(E/(1-(nu)^2))*(exx+nu*eyy);
        Syy=(E/(1-(nu)^2))*(eyy+nu*exx);
        Sxy=(E/(1+nu))*exy;
        Se=sqrt(Sxx^2-Sxx*Syy+Syy^2+3*Sxy^2);
        %%%Von Mises Criteria
        if Se>Sy
            display('Failure for element' )
            display('Ely=')
            display(ely)
            display('Elx=')
            display(elx)
            display('Stress Se=')
            display(Se)
        else
            display('Stress under yield for element' )
            display('Ely=')
            display(ely)
            display('Elx=')
            display(elx)
            display('Stress Se=')
            display(Se)
        end
    end
end
end
end
%%%%%% OPTIMALITY CRITERIA UPDATE
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
    lmid = 0.5*(l2+l1);
    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-
dc./lmid)))));
    if sum(sum(xnew)) - volfrac*nelx*nely > 0;
        l1 = lmid;
    else
        l2 = lmid;
    end
end
end

```



```

##### MESH-INDEPENDENCY FILTER
function [dcn]=check(nelx,nely,rmin,x,dc)
dcn=zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        sum=0.0;
        for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
            for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
                fac = rmin-sqrt((i-k)^2+(j-l)^2);
                sum = sum+max(0,fac);
                dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
            end
        end
        dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
    end
end
##### FE-ANALYSIS
function [U]=FE(nelx,nely,x,penal)
[KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
    for ely = 1:nely
        n1 = (nely+1)*(elx-1)+ely;
        n2 = (nely+1)* elx +ely;
        edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1;
2*n1+2];
        K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
    end
end
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
%F(2*(nely+1)*(nelx+1)-2*nely,1)=-1;
%Force at the upper right corner of the beam
%F((nely+1)*nelx+2,1) = -1; %for even nelx
%Force at the center of the beam
F([2*(nely+1)+2:2*(nely+1):2*(nely+1)*(nelx+1)-2*nely],1)=-1/nelx;
% for uniformly distributed load
%fixeddofs = union([1:2*(nely+1)], [2*(nely+1)*nelx-
1:2*(nelx+1)*(nely+1)]);
%for fixed ends at both sides
%fixeddofs=union([2*(nely+1)], [2*(nelx+1)*(nely+1)-1,
2*(nelx+1)*(nely+1)]);
%scrolling at the left side and joint at the right side
fixeddofs = [1:2*(nely+1)];
%from fixed end at the left side of the beam
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
##### ELEMENT STIFFNESS MATRIX
function [KE]=lk
E = 1.;
nu = 0.3;
k=[ 1/2-nu/6    1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
-1/4+nu/12 -1/8-nu/8    nu/6    1/8-3*nu/8];
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)

```

k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)  
k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)  
k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

## Program for the uncertain load case- 1 load

```

function top2(nelx,nely,volfrac,penal,rmin,w,vmean,vsigma)
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
load w4.dat %% the name of the file can be changed according to the
type of problem
load x4.dat
m=w4(1);
xd=x4(2:m+1);
wd=w4(2:m+1);
% START ITERATION
while change > 0.01
    mean1=0;
    mean2=0;
    dmean1=0;
    dmean2=0;
    loop = loop + 1;
    xold = x;
    % FE-ANALYSIS
    c=zeros(m,1);
    for n=1:m
        [U]=FE(nelx,nely,x,penal,xd,n,vmean,vsigma);
        %%OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
        [KE] = lk;
        for ely = 1:nely
            for elx = 1:nelx
                n1 = (nely+1)*(elx-1)+ely;
                n2 = (nely+1)* elx +ely;
                Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2;
2*n1+1;2*n1+2],1);
                c(n) = c(n) + x(ely,elx)^penal*Ue'*KE*Ue;
                dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
            end
        end
        mean1=mean1+wd(n)*c(n);
        mean2=mean2+wd(n)*power(c(n),2);
        dmean1=dmean1+wd(n)*dc;
        dmean2=dmean2+(2*wd(n)*c(n)*dc);
    end
    mean1=mean1*(1/sqrt(pi));
    mean2=mean2*(1/sqrt(pi));
    dmean1=dmean1*(1/sqrt(pi));
    dmean2=dmean2*(1/sqrt(pi));
    sigma=sqrt(mean2-power(mean1,2));
    dsigma=(dmean2-2*mean1*dmean1)/(2*sigma);
    Frob=w*mean1+(1-w)*sigma;
    dFrob=w*dmean1+(1-w)*dsigma;
    % FILTERING OF SENSITIVITIES
    [dFrob] = check(nelx,nely,rmin,x,dFrob);
    % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
    [x] = OC(nelx,nely,x,volfrac,dFrob);
    % PRINT RESULTS
    change = max(max(abs(x-xold)));
    disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',Frob)
...
' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...

```

```

    ' Ch.: ' sprintf('%6.3f',change ))
% PLOT DENSITIES
    colormap(gray); imagesc(-x); axis equal; axis tight; axis
off;pause(1e-6);
end
disp(['It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',Frob)...
    ' mean1: ' sprintf('%10.4f',mean1) ' sigma:'
sprintf('%10.4f',sigma)])
##### OPTIMALITY CRITERIA UPDATE
#####
function [xnew]=OC(nelx,nely,x,volfrac,dFrob)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
    lmid = 0.5*(l2+l1);
    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-
dFrob./lmid)))));
    if sum(sum(xnew)) - volfrac*nelx*nely > 0;
        l1 = lmid;
    else
        l2 = lmid;
    end
end
end

##### MESH-INDEPENDENCY FILTER
#####
function [dcn]=check(nelx,nely,rmin,x,dFrob)
dcn=zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        sum=0.0;
        for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
            for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
                fac = rmin-sqrt((i-k)^2+(j-l)^2);
                sum = sum+max(0,fac);
                dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dFrob(l,k);
            end
        end
        dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
    end
end
end

##### FE-ANALYSIS
#####
function [U]=FE(nelx,nely,x,penal,xd,n,vmean,vsigma)
[KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
    for ely = 1:nely
        n1 = (nely+1)*(elx-1)+ely;
        n2 = (nely+1)* elx +ely;
        edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1;
2*n1+2];
        K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
    end
end
end

% DEFINE LOADS AND SUPPORTS
F(2*(nely+1)*(nelx+1)-2*nely,1)=vmean+sqrt(2)*vsigma*xd(n);
fixeddofs = [1:2*(nely+1)];
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING

```

```

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
%%%%%%%%% ELEMENT STIFFNESS MATRIX
%%%%%%%%%
function [KE]=lk
E = 1.;
nu = 0.3;
k=[ 1/2-nu/6    1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
   -1/4+nu/12 -1/8-nu/8  nu/6      1/8-3*nu/8];
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
                  k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

```

## Program for the uncertain load case-2 loads

```

function
run_top2n(nelx,nely,volfrac,penal,rmin,w,vmean1,vmean2,vsigma1,vsigma2
)
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
load w2.txt %% the name of the file can be changed according to the
type of problem
load x2.txt
m=w2(1);
xd1=x2(2:m+1);
xd2=x2(m+2:2*m+1);
wd=w2(2:m+1);
% START ITERATION
while change > 0.01
    mean1=0;
    mean2=0;
    dmean1=0;
    dmean2=0;
    loop = loop + 1;
    xold = x;
    % FE-ANALYSIS
    c=zeros(m,1);
    for n=1:m

        [U]=FE(nelx,nely,x,penal,xd1,xd2,n,vmean1,vmean2,vsigma1,vsigma2);
        %%OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
        [KE] = lk;
        for ely = 1:nely
            for elx = 1:nelx
                n1 = (nely+1)*(elx-1)+ely;
                n2 = (nely+1)* elx +ely;
                Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2;
                2*n1+1;2*n1+2],1);
                c(n) = c(n) + x(ely,elx)^penal*Ue'*KE*Ue;
                dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
            end
        end
        mean1=mean1+wd(n)*c(n);
        mean2=mean2+wd(n)*power(c(n),2);
        dmean1=dmean1+wd(n)*dc;
        dmean2=dmean2+(2*wd(n)*c(n)*dc);
    end
    mean1=mean1*(1/pi);
    mean2=mean2*(1/pi);
    dmean1=dmean1*(1/pi);
    dmean2=dmean2*(1/pi);
    sigma=sqrt(mean2-power(mean1,2));
    dsigma=(dmean2-2*mean1*dmean1)/(2*sigma);
    Frob=w*mean1+(1-w)*sigma;
    dFrob=w*dmean1+(1-w)*dsigma;
    % FILTERING OF SENSITIVITIES
    [dFrob] = check(nelx,nely,rmin,x,dFrob);
    % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
    [x] = OC(nelx,nely,x,volfrac,dFrob);
    % PRINT RESULTS
    change = max(max(abs(x-xold)));

```

```

disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',Frob)
...
      ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
      ' Ch.: ' sprintf('%6.3f',change )])
% PLOT DENSITIES
colormap(gray); imagesc(-x); axis equal; axis tight; axis
off;pause(1e-6);

picturename=sprintf('%d_%d_%f_%d_%f_%f_%d_%d_%f_%f_%d.png',nelx,nely,
volfrac,penal,rmin,w,vmean1,vmean2,vsigma1,vsigma2,m);
print('-dpng','-r300',picturename)
end
disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',Frob)...
      ' mean1: ' sprintf('%10.4f',mean1) ' sigma:'
sprintf('%10.4f',sigma)])
%%%%%%%%%% OPTIMALITY CRITERIA UPDATE
%%%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dFrob)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
    lmid = 0.5*(l2+l1);
    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-
dFrob./lmid)))));
    if sum(sum(xnew)) - volfrac*nelx*nely > 0;
        l1 = lmid;
    else
        l2 = lmid;
    end
end
end

%%%%%%%%%% MESH-INDEPENDENCY FILTER
%%%%%%%%%%
function [dcn]=check(nelx,nely,rmin,x,dFrob)
dcn=zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        sum=0.0;
        for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
            for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
                fac = rmin-sqrt((i-k)^2+(j-l)^2);
                sum = sum+max(0,fac);
                dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dFrob(l,k);
            end
        end
        dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
    end
end
end

%%%%%%%%%% FE-ANALYSIS
%%%%%%%%%%
function
[U]=FE(nelx,nely,x,penal,xd1,xd2,n,vmean1,vmean2,vsigma1,vsigma2)
[KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
    for ely = 1:nely
        n1 = (nely+1)*(elx-1)+ely;
        n2 = (nely+1)* elx +ely;
        edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1;
2*n1+2];
        K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
    end
end

```

```

end
end
% DEFINE LOADS AND SUPPORTS
F(2*(nely+1)*(nelx+1)-2*nely,1)=vmean1+sqrt(2)*vsigma1*xd1(n);
%Force at the upper left corner of the beam
F((nely+1)*(nelx+2),1)=vmean2+sqrt(2)*vsigma2*xd2(n);
%Force at the lower center of the beam
fixeddofs = [1:2*(nely+1)];
alldofs    = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
%%%%%%%%%% ELEMENT STIFFNESS MATRIX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [KE]=lk
E = 1.;
nu = 0.3;
k=[ 1/2-nu/6    1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
   -1/4+nu/12 -1/8-nu/8  nu/6    1/8-3*nu/8];
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
                  k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

```





ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ  
ΒΙΒΛΙΟΘΗΚΗ



004000125708