BI-DIRECTIONAL EVOLUTIONARY ALGORITHM FOR VOLUME CONSTRAINED TOPOLOGY OPTIMIZATION OF AXISYMMETRIC SOLIDS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OĞUZ ZİYA TİKENOĞULLARI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2015

Approval of the thesis:

BI-DIRECTIONAL EVOLUTIONARY ALGORITHM FOR VOLUME CONSTRAINED TOPOLOGY OPTIMIZATION OF AXISYMMETRIC SOLIDS

submitted by **OĞUZ ZİYA TİKENOĞULLARI** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Tuna Balkan _____
Head of Department, **Mechanical Engineering**

Prof. Dr. Suha Oral _____
Supervisor, **Mechanical Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Müfit Gülgeç
Mechantronics Engineering Department, Çankaya University _____

Prof. Dr. Suha Oral
Mechanical Engineering Department, METU _____

Prof. Dr. Suat Kadıoğlu
Mechanical Engineering Department, METU _____

Prof. Dr. Serkan Dağ
Mechanical Engineering Department, METU _____

Assist. Prof. Dr. Hüsnü Dal
Mechanical Engineering Department, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: OĞUZ ZİYA TİKENOĞULLARI

Signature :

ABSTRACT

BI-DIRECTIONAL EVOLUTIONARY ALGORITHM FOR VOLUME CONSTRAINED TOPOLOGY OPTIMIZATION OF AXISYMMETRIC SOLIDS

TİKENOĞULLARI, OĞUZ ZİYA
M.S., Department of Mechanical Engineering
Supervisor : Prof. Dr. Suha Oral

September 2015, 71 pages

In this thesis, topology optimization of axisymmetric solids is studied. Analysis of the axisymmetric problem is performed by coding an axisymmetric finite element formulation in association with the optimization code which is based on Bi-Directional Evolutionary Optimization (BESO) method. The optimization method used in this study includes recent improvements to the evolutionary optimization algorithms. These are bi-directional evolution, sensitivity number filtering and sensitivity-time averaging. In the optimization process, mean compliance of the overall structure is minimized while gradually removing material in order to reach volume constraint. Removal of the material is decided according to the strain energy stored within the volume of the finite element. In this study, hard-kill method is used for the element removal. Therefore removed material has no contribution to the stiffness of the structure. In the analysis of the structure, iso-parametric axisymmetric finite element formulation is used.

Both the analysis and optimization codes are run in a successive manner. Sample problems are chosen from the literature and solved with the present optimization method. Results are compared and performance of the method is discussed.

Keywords: Bi-directional, Evolutionary, Topology, Structural, Axisymmetric, Optimization

ÖZ

EKSENEL SİMETRİK YAPILARIN HACİM KISITLAMALI ÇİFT YÖNLÜ EVRİMSEL ALGORİTMALAR İLE OPTİMİZASYONU

TİKENOĞULLARI, OĞUZ ZİYA
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Suha Oral

Eylül 2015 , 71 sayfa

Bu tezde eksenel simetrik yapıların evrimsel optimizasyonu çalışılmıştır. Eksenel simetrik optimizasyon probleminin çözümü, çift yönlü evrimsel optimizasyon metodu (BESO) tabanlı optimizasyon yöntemi ile sonlu elemenlar yöntemi bir arada kullanılarak gerçekleştirilmiştir. Bu çalışmada kullanılan optimizasyon yöntemi son zamanlarda evrimsel optimizasyon yöntemi (ESO) üzerinde yapılan iyileştirmeleri de kapsamaktadır. Bunlar; çift yönlü evrimin eklenmesi, duyarlılık sayılarının filtrelenmesi ve duyarlılık sayılarının zamana göre ortalamasının alınması yöntemleridir. Optimizasyon sürecinde tüm yapının hacmi adım adım azaltılarak hedeflenen hacimsel orana ulaştırılırken yapının eşdeğer esnekliği azaltmaya çalışılmaktadır. Malzemenin boşaltılmasındaki kriter olarak, sonlu elemanın hacminde depolanan gerinim enerjisi kullanılmaktadır. Bu çalışmada malzemenin boşaltılmasında tam boşaltma yöntemi kullanılmaktadır. Bu sebepten, malzeme boşaltıldıktan sonra yapının rijitliği üzerinde hiç bir etkisi

kalmamaktadır. Yapının analizinde eş-parametrik sonlu eleman formulasyonu kullanılmaktadır. Optimizasyon ve analiz kodları, süreç bounca ardışık şekilde çalışmaktadır. Literatürden seçilen problemler sunulan bu optimizasyon yöntemi ile çözülmüştür ve sonuçlar literatürle karşılaştırılırken optimizasyon yönteminin performansı hakkında yorumlamalar yapılmıştır.

Anahtar Kelimeler: Çift Yönlü, Evrimsel, Topoloji, Yapısal, Eksenel Simetrik, Optimizasyon

To my grandfather

Ziya Tikenoğulları

ACKNOWLEDGMENTS

I would like to thank to my advisor Prof. Suha Oral for his constant support during my thesis study. His toleration always kept my motivation high and his deep knowledge guided me when I had hard times at my studies. I must admit that I would not be able to complete my study without his guidance.

My sincere thanks goes to my friends Halilcan Toksöz, Ali Çağatay Çobanoğlu and Deniz Ayhan. They have helped me have the ideal working environment for my study, always guided me with their deep knowledge and supported me emotionally to keep motivated through tough times. They have contributed to a part which is not written in the following pages.

Lastly I would like to express my deep appreciation to my family, for supporting me with my decisions all the time and supporting me not only in my studies but in life.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

C	Mean compliance of the structure
V	Volume of the structure
V^*	Target final volume of the structure
K	Global stiffness matrix
K_i	Stiffness matrix of i^{th} element
N^e	Shape function
E	Young's modulus
u	Global nodal displacements vector
f	Global nodal force vector
x	Design parameters
r, z	Coordinate axes
w	Weight factor
r_{min}	Sensitivity number filtering radius
$e_{rr}, e_{zz}, e_{\theta\theta}, e_{rz}$	Strain components
u_r, u_z	Displacement components
ξ, η	Element coordinates
α	Sensitivity number
α_{add}^{th}	Threshold sensitivity number for element addition
α_{del}^{th}	Threshold sensitivity number for element removal
ν	Poisson's ratio
τ	Convergence value
Ω	Element domain
Γ	Element boundary
$\mathbf{N}^e$	Shape function matrix
$\mathbf{B}$	B matrix containing shape function gradient information
$\mathbf{E}$	Material stiffness matrix
$\hat{\mathbf{t}}$	Traction vector
J	Jacobian matrix

AR_{max}	Maximum volume addition ratio
GUI	Graphical user interface
$2D$	Two dimensional
$3D$	Three dimensional
ESO	Evolutionary structural optimization
$BESO$	Bi-directional evolutionary structural optimization
CAD	Computer aided drawing
$SIMP$	Solid isotropic material with interpolation
ER	Evolutionary ratio

CHAPTER 1

INTRODUCTION

In structural engineering, optimization methods have a wide range of possible applications and structural optimization gains popularity with the increasing computational power. Optimization can be used in different steps of the design process. Structural optimization can be separated into three classes; size optimization, shape optimization and topology optimization. Size optimization is the search for the optimum thickness of predefined members, whereas shape optimization is the search for the optimum shape of the predefined boundaries of members. Topology optimization is not constrained by predefined members or boundaries; searches for the optimum shape and layout of cavities within a structure (graphical illustration of three classes is given in figure 1.1). Therefore topology optimization is the most challenging and the most rewarding optimization class. Topology optimization is suitable for the early design phases. It helps the designer to understand the design domain better and explore the possible efficient design options. Generally in a design process, topology optimization is followed by shape and size optimizations. Therefore, topology optimization is not excepted to result in a final product, instead it is expected yo give a rough idea about the most efficient topology.

In the recent years many topology optimization methods were developed such as ESO/BESO (Xie and Steven, 1993 [42]; Huang and Xie, 2007 [16]), homogenization method (Bendsøe and Kikuchi, 1988 [2]; Bendsøe and Sigmund, 2003 [4]), SIMP method (Bendsøe, 1989 [3]; Zhou and Rozvany, 1991 [45]; Rietz, 2001 [29]; Bendsøe and Sigmund, 2003 [4]) and level set method (Sethian and

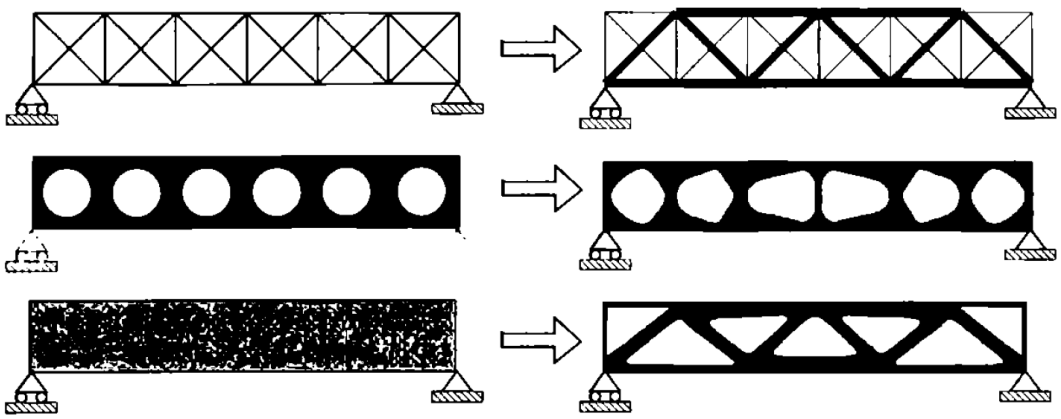

Figure 1.1: Visualization of classes of structural optimization [4] (Top to bottom: Size optimization, shape optimization and topology optimization)

Wiegmann, 2000 [32]; Wang et al., 2003 [40]; Allaire et al., 2004 [1]; Wang et al., 2004 [41]). Beyond these methods, evolutionary structural optimization (ESO and later developed into BESO) method is one of the efficient and effective methods. Following years of the establishment of the evolutionary optimization algorithm, many studies and critics are made on this topic. These studies pointed out some shortcomings of this method thus, lead to various improvements over the original method. With these improvements BESO method has become even more effective which is one of the reasons of this study.

In the literature there is a wide range of applications of topology optimization methods. However, most of them focus on either 2D or 3D problems. Applications of axisymmetric topology optimization are limited in the literature. Therefore this is another motivation for the topic of this study.

Objective of this thesis is to apply Bi-directional Evolutionary Structural Optimization method together with some of these important improvements in order to solve optimization problem of an axisymmetric solid. For this purpose, different case problems are solved and some are compared with the solutions from literature.

1.1 Problem Definition

In this study, topology of an axisymmetric solid is optimized. Optimization procedure is based on the BESO method with slight modifications that are discussed in the following chapters. In each iteration of the optimization algorithm, a finite element analysis is performed. At the end of these successive iterations the topology is expected to converge to an optimum. After convergence, the resulting structure is accepted as the optimized structure.

In the process of seeking for the optimum, mean compliance of the structure is minimized while structure gradually approaches to the volume constraint. When the structure is brought to the final target volume, optimization process continues until convergence of the mean compliance value is obtained. This problem type is called the "Stiffness optimization with a volume constraint". On the other hand, in some practical applications stress constraints may be important. In this case the minimum volume (or mass) of the structure is searched, while structure is able to withstand the defined loads. This is the "Mass optimization with stress constraints".

Major topology optimization methods are originally based on mean compliance minimization, which can be considered as the simplest type of formulation and a natural starting point as its solution reflects many of the fundamental issues in the field (Bendsøe and Sigmund, 2003[4]). Although stiffness optimization can be chosen due to mathematical simplicity (Jouve, 2014[20]) or its computational efficiency (Holmberg, 2013[15]); it is sufficient for the needs of topology optimization for most of the applications. In practical problems, stiffness optimization is used in order to identify the load paths. Stress constraints (and other design constraints) are implemented in the shape and size optimization steps (Krog et al., 2002[22]). Hence, stiffness minimization formulation is intensively used in stress constrained industrial applications.

On the other hand, some researchers have applied topology optimization methods to stress constrained mass minimization (e.g., Duysinx and Bendsøe, 1998[12]; Shim and Manoochehri, 1997[33]). These applications help the

designers achieve a conceptual design which is closer to the final design[15]. In this thesis, minimization of mean compliance subject to volume constraint is studied. Stress constraints and mass minimization is out of the scope of this study. However mass optimization with stress constraints is recommended for future researches and discussed in section5.3.

Resulting optimized topologies do not completely define the final shape of the optimized structure. Objective of the optimization process is to give a rough idea about the optimum topologies. In order to obtain the well-defined shape of a structure, results of the topology optimization may be converted into CAD models and they can be further optimized by applying shape and size optimization methods and they can be modified regarding the manufacturability concerns. However, these issues are out of the scope of this study. In this thesis objective is not to obtain an end product; instead, focus is only on the topology optimization procedure.

1.2 Literature Survey

Beginning with the late '80s, numerical methods for topology optimization has been studied widely; as in the example of the paper of Bendsøe and Kikuchi (1988)[2] where they presented the homogenization method. Similar to homogenization method in some aspects, ESO has been first developed by Xie and Steven[42] and they presented the first ESO method in the 1993 in their paper. In the beginning, it was mostly criticized as an intuitive approach to optimization, however its mathematical theory is explained later by Tanskanen (2002[38]). In his study, BESO is considered to have a mathematical theory and it is shown that this method minimizes mathematical expression of the compliance-volume product .

In 1997 Xie and Steven [43] summarized the early developments of the ESO method in their book. In the following years, a rapid development is observed in this area. At the same time there was concerns about the effectivity of the evolutionary methods. Different researchers pointed the shortcomings of the

ESO in their studies. Most of the earlier studies about evolutionary methods neglected those problems. Some of those problems were convergence, local optimum, checkerboard and mesh dependency problems.

One major shortcoming of the ESO is that it is not able to recover prematurely deleted elements. This may lead to local optimum solutions and makes it necessary to use small evolutionary ratios. Therefore more number of iterations are needed and computational cost increases. To overcome this problem, bi-directional evolutionary method was introduced by Yang et al. (1999)[44]. However this method was using separate addition and removal ratios, which makes the solution dependent on the selections of the parameters. In 2007, Huang and Xie[16] introduced a new approach to BESO. In new BESO method, both material addition and removal are treated together which increases the overall performance of the method and makes the algorithm more logical.

Another problem was addressed by Sigmund (1997)[34] and Sigmund and Peterson (1998)[35] by introducing the sensitivity number filtering scheme. This method averages the sensitivity numbers of the elements using the nearby elements as it is described in detail in the following sections. Assuming that the filtering diameter is chosen correctly, this method helps eliminate mesh dependency and checkerboard problems. Another solution to the checkerboard problem was put forward by Li et al (2001)[23] which is the sensitivity number smoothing scheme. However this method is not effective against the mesh dependency problem. Therefore the filtering scheme is implemented to the BESO method in this study.

Huang and Xie (2007)[16] has addressed the convergence problem later in their study by applying history averaging to the elemental sensitivity numbers. The convergence problem arises from the fact that the design variables are discrete, therefore objective function encounters big jumps between the iterations leading to chaotic convergence curves. In order to avoid this, sensitivity number changes should be smoothened. This is achieved by averaging the current sensitivity numbers with the ones from the previous iteration. In this way, a more stable process is obtained and convergence is achieved.

Another shortcoming of the ESO/BESO is the one shown by Zhou and Rozvany (2001)[46] which is yet to be solved. In that example of low stressed elements near the support are removed by the algorithm which leads to a much worse design. In order to prevent this problem, a finer mesh should be used or mesh refinement between the optimization steps can be a remedy. However, there will be always a stress value that will lead to this problem (Rozvany 2009[30]). Therefore one should be aware of this problem when using evolutionary optimization methods. Evolution process is monitored in this study, order to determine whether element removal leads to compliant structures.

CHAPTER 2

OPTIMIZATION PROBLEM FORMULATION

In this chapter, a detailed explanation of the optimization method is made. Firstly the general methodology of the BESO is given and in the following chapters, its components are explained. Design parameters, objective function, constraints and convergence criterion are explained in the way they are used in this study. As discussed before, sensitivity number averaging and filtering methods are implemented to the classical BESO method and they are explained in detail regarding the mathematical aspects.

2.1 Bi-directional Evolutionary Structural Optimization

Xei and Huang (2010)[17] define the topology optimization procedure as to find the topology of a structure by determining for every point in the design domain if there should be material (solid element) or not (void element). Because it is impossible to determine this condition for infinite number of points in the structure, the design domain is discretized into finite elements and these elements are determined to be solid or void. While determining material presence, we need to change material from solid to void or from void to solid. In order to perform such operations, in this study BESO is used as the optimization methodology.

Bi-directional Evolutionary Structural Optimization method (BESO) is a variety of another topology optimization method known as Evolutionary Structural Optimization method (ESO). It allows material addition and removal at the same time, whereas in ESO only the material removal is allowed. By adding

material addition capability, BESO is obtained.

As discussed earlier, a few improvements are implemented in BESO and they are used in this study also. They are sensitivity filtering scheme and sensitivity number averaging. Together with these improvements, a generalized flowchart of the BESO method can be represented as in the figure 2.1:

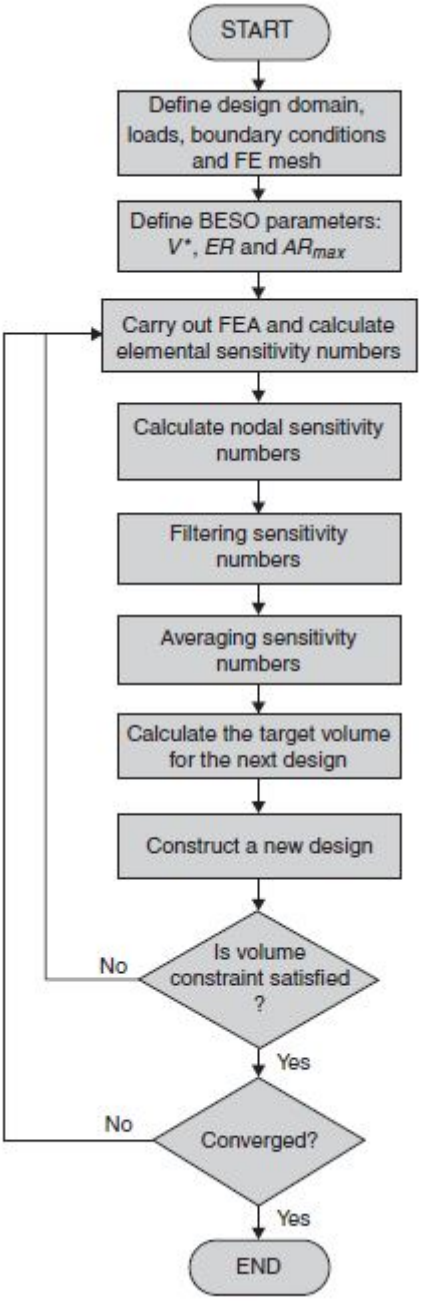

Figure 2.1: BESO flowchart

2.2 Design Parameters

In the topology optimization, the objective is to determine if material is needed, for every point in the design domain. Optimized structure is searched by changing element presence for these points. Therefore, the design parameter is the elements themselves. Depending on the topology optimization method, design parameters can be continuous or discrete. Intermediate elements are used in some topology optimization methods like SIMP (uses continuous intermediate element properties) and soft-kill BESO (uses discrete intermediate element properties). In these methods elements are not either solid or void but they can have intermediate values. These values are usually obtained by multiplying the element stiffness values by a penalty coefficient.

In this study, hard-kill BESO method is used where the design parameters are discrete and can have two different values, either 0 or 1. Therefore there is no intermediate elements or so called gray regions. However removing the elements by simply changing the stiffness values to 0 may introduce some problems. These zero elements may lead to free degree of freedoms therefore lead to singularities while computing the displacements from force and stiffness matrices. In order to prevent this problem, elements are removed by changing from 1 to 10^{-6} as suggested by Hinton and Sienz (1995)[14]. The value of 10^{-6} is small enough and close to zero, however it is large enough to get rid of the singularity problem computationally. In the results, these elements are shown as they were zero elements.

2.3 Objective Function

Mean compliance is chosen as the objective function in this study. Therefore mean compliance is minimized throughout the optimization process. As the element removal criterion, strain energy based sensitivity numbers are used. In the following chapters these are explained regarding the mathematical aspects.

2.3.1 Sensitivity Numbers

Initially, Evolutionary Structural Optimization method (ESO) was widely used with removal ratios (RR) based on the stress levels. In this method, generally the optimization continues until a desired uniform stress distribution is obtained (for example, all stress values are within the 75% of the stress of maximum stressed element). However, such a solution may or may not exist depending on the problem. In BESO, generally removal and addition ratios are determined according to the target volume of the corresponding step (not the maximum stress). However element removal criterion (sensitivity numbers) may be based on stress levels or strain energy levels. In this study, the sensitivity numbers are based on the strain energy levels. However it should be noted that either choosing the sensitivity numbers based on stress levels or strain energy levels often lead to similar topologies (Huang and Xie 2010[17]). In this study, sensitivity numbers are used as:

$$\alpha_i^e = \frac{1}{2} u_i^T K_i \, u_i / V_i \tag{2.1}$$

where α_i^e represents the strain energy density of the i^{th} element. Please note that V_i (volume of the i^{th} element) changes because of the varying radial positions of the elements in the axisymmetric case. Also note that in order to overcome some problems of BESO, the sensitivity numbers are not used as they are found as above. Sensitivity numbers are manipulated by methods such as sensitivity number filtering or history averaging which are discussed in the following sections.

2.3.2 Mean Compliance

ESO was first introduced as a method that was inspired from the nature. It was considered as an intuitive way to optimization by many researchers and criticized for lacking mathematical foundations. Tanskanen (2002)[38] has shown the mathematical theory behind the ESO method and shown that ESO minimizes the mean compliance-volume product ($C \cdot V$) of the structure. However this

minimization leads to zero volume for most of the structures (Rozvany 2002)[31]. In order to prevent this problem, formulation was changed to mean compliance minimization by setting a target final volume (V^*). In the new formulation, the volume fraction of the structure is brought to a pre-defined value V^* while keeping the mean compliance at minimum. Here the mean compliance is defined as:

$$C = \frac{1}{2} f^T u \tag{2.2}$$

and it is the objective function in the new BESO method. In the above equation, f is the global force vector and u is the global displacement vector.

2.4 Constraints

The only constraint imposed to the optimization problem is the equality constraint of the volume fraction, that is V^*. Depending on the optimization problem, stress constaint may be applied in the optimization formulation. Inclusion of stress constraints leads to a completely different formulation of sensitivity numbers and handling stress constraints introduces various problems. Because of these reasons, the work in this thesis is limited to volume constrained optimization only.

In this study, initial volume fraction is chosen as 100% and volume fraction is decreased gradually in each step. When the volume fraction (V) is satisfied in a particular step (k), it is decreased by the evolution ratio (ER) as given below and process continues with the next iteration ($k+1$).

$$V_{k+1} = V_k \, (1 - ER) \qquad (k = 1, 2, 3, \ldots) \tag{2.3}$$

These steps are applied successively until the V^* is reached. In this way, the final volume fraction is set equal to the pre-defined value V^*. Volume constraint can be shown mathematically as:

$$V^* - \sum_{i=1}^{N} V_i x_i = 0 \tag{2.4}$$

In the equation2.4, V_i is the volume of the i^{th} element and x_i is the presence value of the i^{th} element that is, $x_i = 0$ for void elements and $x_i = 1$ for solid elements.

One must note that equality constraints cannot be exactly satisfied for most of the time. This brings the necessity to introduce a tolerance value or another method that helps to accept a value which is close enough to the constraint. In this study, the number of solid elements that exactly satisfies the volume constraint is calculated. The calculated number would be a real number in general. This number is rounded to the nearest integer and the actual number of solid elements that roughly satisfies the volume constraint is found. Details of this process is discussed in the section 2.7.

The only constraint other than volume constraint, is used in case problem 3 (section) 4.3). In this problem design domain is constrained by preventing removal of elements in specified regions. Details of this constraint and its method of application is discussed in section 4.3.

2.5 Sensitivity Number Filtering

Evolutionary methods are used together with finite element analysis in general. Finite element method bring many advantages to the optimization process and it is suitable to the concept of topology optimization. However, the sensitivity numbers of the low order finite elements can become discontinuous across the element boundaries (Jog and Haber 1996[19]). These discontinuities cause the problem named checkerboard problem. The reason for checkerboard problem is that; in such a finite element discretization introducing more holes into the structure makes the structure more efficient and the optimization algorithm ends up with the so called checkerboard pattern.

In order to overcome the checkerboard pattern, various researchers have come

up with different solutions over the years. Using polygonal elements (Pereira et al 2010 [37]), perimeter control method (Jog 2002[18]), averaging sensitivity numbers with neighboring elements (Li et al. 2001 [23]) and sensitivity number filtering scheme (Sigmund and Petersson 1998[35]) are the examples of such solutions.

In addition to checkerboard problem, in evolutionary methods the mesh dependency problem is also encountered frequently. This problem is defined as having a qualitatively different optimal topology as the mesh refinement changes. If a correct methodology is followed, the same topology should be obtained with a finer mesh, however only with more accurate definition of the boundaries.

Two of the above mentioned methods; namely, perimeter control method and the sensitivity number filtering method are also shown to be able to solve the mesh dependency problem. Because of the easy implementation advantage, sensitivity number filtering scheme is preferred in order to implement into the BESO in this study. With the help of this method, both the checkerboard problem and the mesh dependency problem are overcome. The procedure of sensitivity number filtering is performed in two steps:

1. *Computing nodal sensitivity numbers*

   Each node is assigned a value by averaging the sensitivity numbers of the elements those are connected to that node. Nodal values are calculated according to the following formula:

$$\alpha_j^n = \sum_{i=1}^{M} w_i \alpha_i^e \tag{2.5}$$

   where α_j^n is the nodal sensitivity number of the j^{th} node and α_i^e is the sensitivity number of the i^{th} element. M is the total number of elements connected to the j^{th} node. w_i is the weight factor of each element and calculated as follows:

$$w_i = \frac{1}{M-1}\left(1 - \frac{r_{ij}}{\sum_{i=1}^{M} r_{ij}}\right) \tag{2.6}$$

where r_{ij} is the distance between the center of i^{th} element and the j^{th} node. In this study, the finite element discretization is made by uniform sized mesh therefore nodes are in equal distance to the connected elements. For example, if there are four elements connected to a node, the weight factor (w_i) simply becomes 0.25 for each element.

2. *Projecting nodal sensitivity numbers back onto elements*

   After computing the nodal sensitivity numbers, they are used to gather back elemental sensitivity numbers. For this purpose, firstly a filtering radius is selected. This filtering radius can be visualized on a generic mesh as given in the figure 2.2.

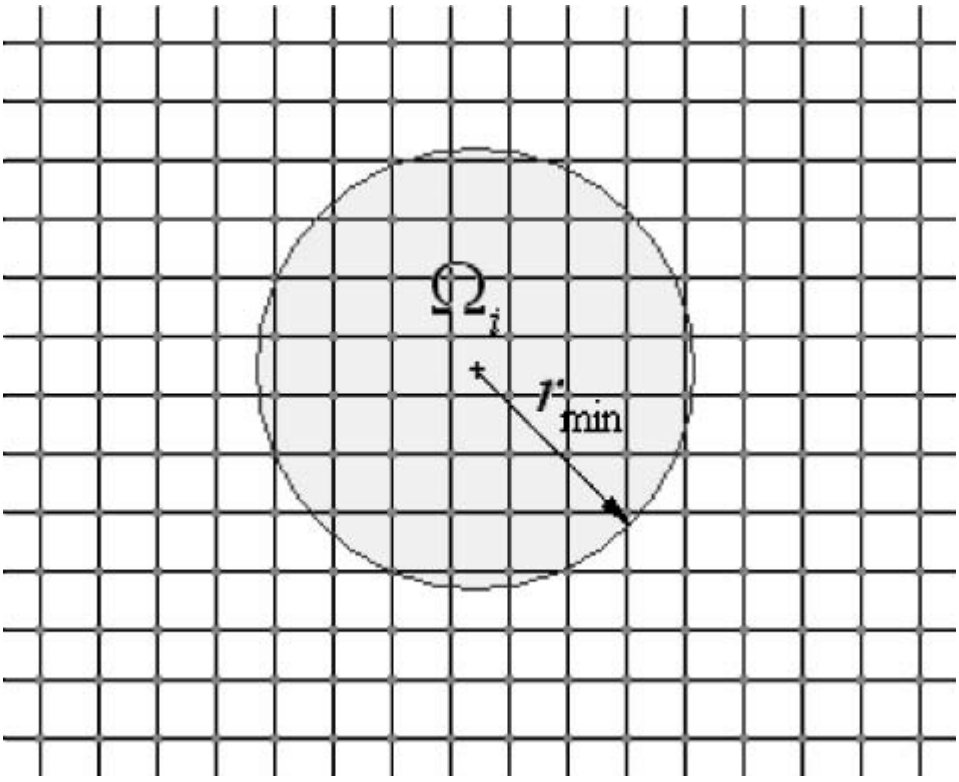

Figure 2.2: Visualization of sensitivity number filtering radius

Also note that this filtering radius ($r_{\min}$) is independent of the mesh size. Only the nodes within $r_{\min}$ distance to an element are used in

the computation of the element sensitivity numbers. Additionally, closer nodes have larger contribution to the related elements' sensitivity numbers, through imposed weight factors. This can be mathematically shown as follows:

$$w(r_{ij}) = r_{\min} - r_{ij} \qquad (j = 1, 2, \ldots, K) \tag{2.7}$$

where, K is the total number of nodes within $r_{\min}$ distance to the center of i^{th} element and r_{ij} is the distance between the center of i^{th} element and the j^{th} node. $w(r_{ij})$ is the weight factor of a node. Negative weight factors imply distant nodes which are not included in the computation. Finally, elemental sensitivity numbers can be computed as follows:

$$\alpha_i = \frac{\sum_{j=1}^{K} w(r_{ij})\alpha_j^n}{\sum_{j=1}^{K} w(r_{ij})} \tag{2.8}$$

Using these sensitivity numbers in the addition and removal operations help preventing the checkerboard pattern and mesh dependency problem.

2.6 Sensitivity Number Averaging

As discussed in the previous sections, the design parameters (element presence conditions) are assumed to be able to have discrete values, which are either 1 or 0. Because of this reason sensitivity numbers of elements change suddenly from one step to the other. Effects of this discrete nature can also been observed from discontinuous trend of the objective function. An element removal-addition operation results in sudden changes in sensitivity number and therefore leads to dramatic changes in the topology in the next removal-addition process. This gives way to a chaotic optimization process and negatively affects the convergence to an optimum.

Against this problem, Huang and Xie (2007)[16] has suggested averaging sensitivity numbers of the current step with that of the previous step as follows:

$$\alpha_i = \frac{\alpha_i^k + \alpha_i^{k-1}}{2} \tag{2.9}$$

Where α_i is the averaged sensitivity numbers for the current iteration, α_i^k is the sensitivity numbers from the current analysis and α_i^{k-1} is these sensitivity numbers from the previous iteration. α_i^k is the filtered sensitivity numbers, as discussed in the filtering scheme section.

Using the mentioned methodology, sensitivity number contains the whole history information from previous steps and therefore smoother sensitivity number changes are obtained. Smoother sensitivity number history stabilizes the optimization process and increase the speed of convergence. The effect of this stabilization can be seen from the evolution of the objective function (i.e. mean compliance). Convergence trend of the objective function is given in the figure 2.3, with and without the sensitivity number averaging method.

2.7 Element Addition-Removal

Element addition and removal process is the most important part of the evolutionary process. Everything else is done in order to decide the elements to be removed and the ones to be added. Having determined filtered and averaged sensitivity numbers, addition-removal algorithm performs addition and removal operations based on sensitivity numbers and also taking maximum addition ratio ($AR_{\max}$) and element volumes into account.

The algorithm follows the following steps:

1. All elements (solid and void elements together) are sorted according to their sensitivity numbers. Choosing the ones with higher sensitivity numbers, a design is composed which most closely approximates the volume constraint. According to this design, the total volume of elements needed to be added is calculated. If the ratio of this volume to the total volume is less than the $AR_{\max}$, the design is considered as valid and addition/removal process concludes. If this ratio is higher than the

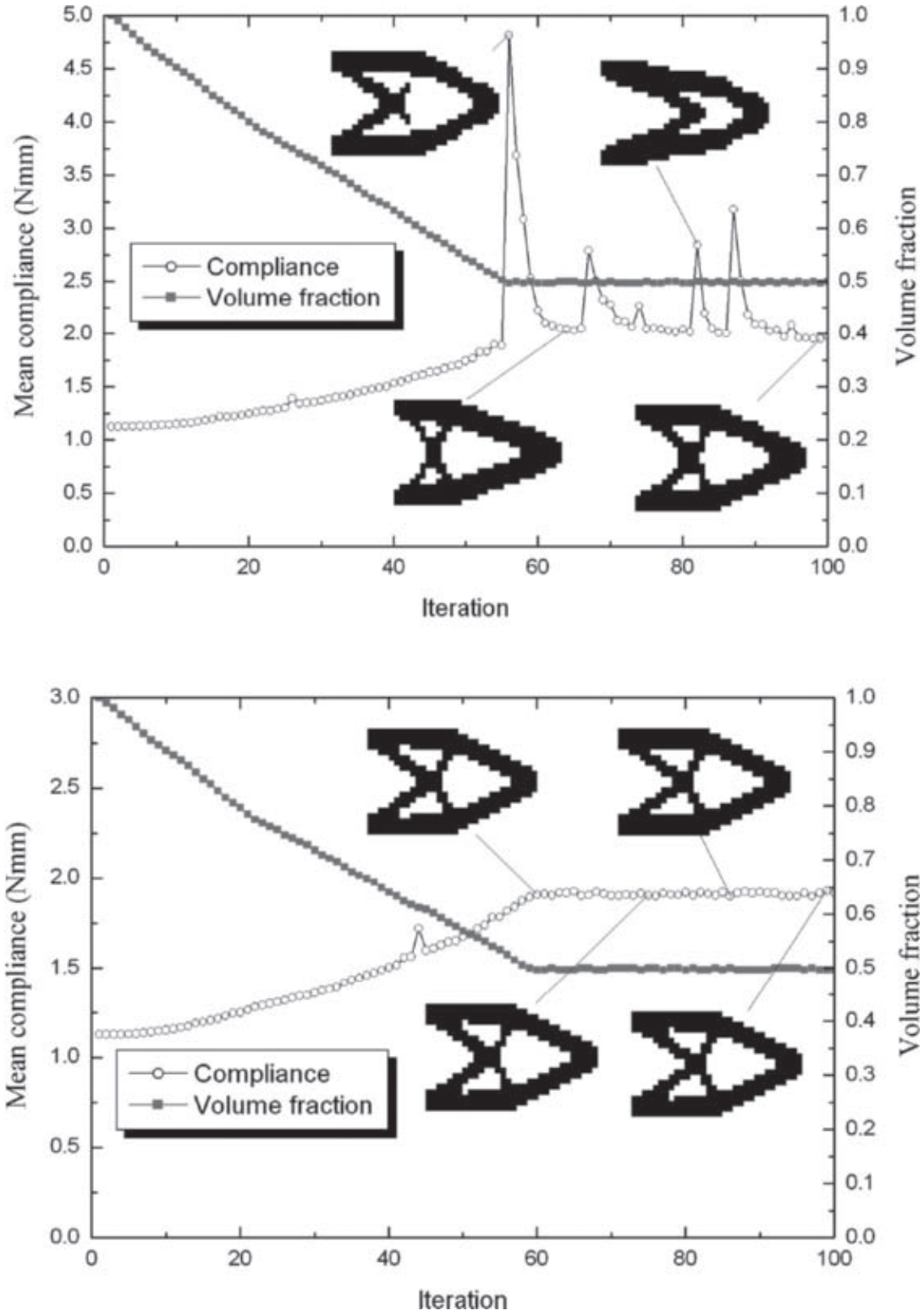

Figure 2.3: Affect of sensitivity number averaging on the evolution of mean compliances [16]

$AR_{\max}$, algorithm continues with the step 2.

2. Void elements with highest sensitivity numbers are turned into solid elements until the volume of these elements reach the limiting volume determined by the $AR_{\max}$. At this point, the composed design has excessive volume because only the element addition operation is performed.

3. After the step 2, element removal operation should be performed in order to approximate the volume constraint (V_k of the k^{th} iteration). For this purpose, solid elements are sorted according to ascending order of sensitivity numbers. Beginning with the ones with smallest sensitivity values, solid element are turned into void until volume constraint is reached. At the end of this step, addition-removal operations finalize and a valid design is obtained.

Process given above includes differences from the method given by the Huang and Xie (2010), they have based the addition-removal steps on so-called threshold sensitivity numbers[17]. However, as a difference, an axisymmetric solid is optimized in this study. Therefore addition-removal algorithm is preferred as given above. Although both algorithms have the same underlying idea, this method is considered as a more logical way when handling elements with varying volumes. Additionally, a better performance is obtained from this algorithm, in terms of approximating the volume constraint more closely.

2.8 Convergence Criterion

As the evolutionary process continues, target volume of specific iteration step gradually approaches to the global volume constraint (V^*). Once target volume becomes equal to the volume constraint ($V_k = V^*$), optimization process continues for a number of iterations until convergence is achieved. In this study, convergence is decided according to the value of the objective function. Error value is computed by taking the N number of previous mean compliances into account. If this error value becomes less than a prescribed

value, convergence criterion is satisfied. Exact calculation of the error can be presented mathematically as below:

$$\text{error} = \frac{\left| \sum_{i=1}^{N} C_{k-i+1} - \sum_{i=1}^{N} C_{k-N-i+1} \right|}{\sum_{i=1}^{N} C_{k-i+1}} \tag{2.10}$$

and convergence criterion is:

$$\text{error} \leq \tau \tag{2.11}$$

This criterion requires the evolutionary process to reach a specific topology and concentrate iterations around this topology. This is considered as a sign of convergence.

As the above criterion is satisfied, the whole optimization process terminates and the final design is taken as the optimum design.

CHAPTER 3

FINITE ELEMENT FORMULATION

In the following chapters, finite element method is explained in detail as it is used specifically in this study. The reasonings behind the selected methods are discussed while mathematical aspects are explained in detail. In the following sections, topics of element selection, element formulation, mesh generation and integration methodology are explained.

3.1 Element selection

Elements used in the finite element analysis are selected regarding a few aspects. Firstly, the elements should be axisymmetric elements (also known as ring elements) because of the axisymmetric design domain assumption in this study. Sketch of such an element is given in below (figure 3.1).

Secondly, the element shapes should be suitable for axisymmetric topology optimization formulation. For this reason, quadrilateral element is chosen because it is easy to formulate in cylindrical coordinates, application of boundary conditions are simple and it is more suitable for the rectangular design domain used in this study. Thirdly, iso-parametric formulation is used in the element formulation. The advantage of the iso-parametric formulation is that it allows use of the same shape functions in order to define displacements and the geometries of the elements, therefore provides simplicity to the finite element formulation. Lastly the shape functions are chosen to be linear as it is the case for the most finite element applications and it is adequate for the purposes of

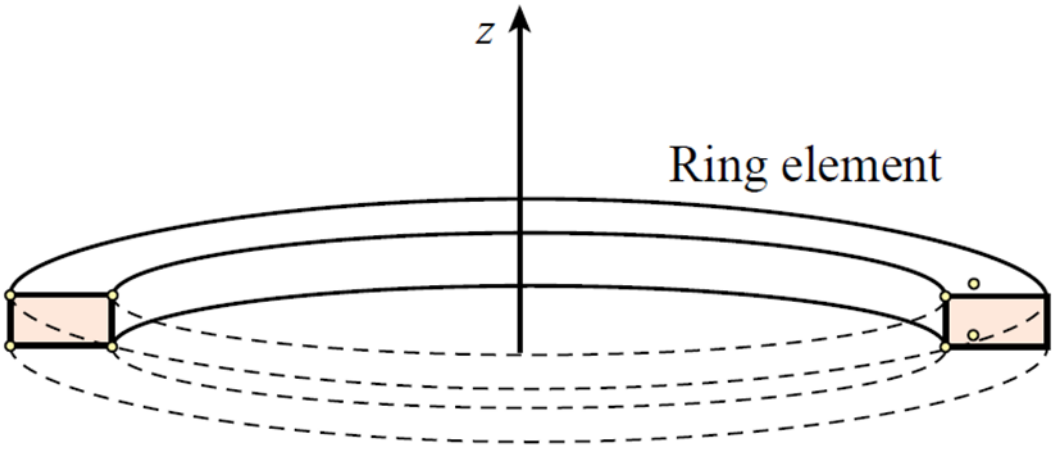

Figure 3.1: Graphical representation of an axisymmetric quadrilateral ring element

this study also.

3.2 Element Formulation

In order to use finite elements in the analysis of the structures, behavior of these elements should be determined by element formulation. This formulation is performed considering that the element is quadrilateral iso-parametric ring element with linear shape functions. Formulations below are compiled mostly from Felippa's (2004)[13] lecture notes.

The formulation should begin with defining the shape functions. Because a quadrilateral element has four nodes, four different linear shape functions should be defined which satisfy the following relation at any point in the element domain.

$$1 = \sum_{i=1}^{4} N_i^e \tag{3.1}$$

Here the index i represents node number and takes values up to 4 because there are 4 nodes in a quadrilateral element. Keeping in mind that the shape functions should be linear, these can be expressed in terms of ξ and eta as follows:

$$N_i^e = \tfrac{1}{4}(1-\xi)(1-\eta) \tag{3.2}$$

$$N_i^e = \tfrac{1}{4}(1+\xi)(1-\eta) \tag{3.3}$$

$$N_i^e = \tfrac{1}{4}(1+\xi)(1+\eta) \tag{3.4}$$

$$N_i^e = \tfrac{1}{4}(1-\xi)(1+\eta) \tag{3.5}$$

As previously stated in section 3.1, these shape functions define both element geometries and displacement of any point on that element. That means, these relations can be written together with equation 3.1 as follows:

$$\begin{bmatrix} 1 \\ r \\ z \\ u_r \\ u_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ r_1 & r_2 & r_3 & r_4 \\ z_1 & z_2 & z_3 & z_4 \\ u_{r1} & u_{r2} & u_{r3} & u_{r4} \\ u_{z1} & u_{z2} & u_{z3} & u_{z4} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \\ N_4^e \end{bmatrix} \tag{3.6}$$

Where r_n, z_n, u_{rn}, u_{zn} are the nodal position and displacement values. In our case $n = 4$ because there are 4 nodes in the quadrilateral elements. As an alternative to the equation 3.6, displacement relations can be written with another form of shape function matrix ($\mathbf{N}$) and nodal displacements matrix ($\mathbf{u}^e$) as follows:

$$\mathbf{u} = \mathbf{N}^e \mathbf{u}^e \tag{3.7}$$

where,

$$\mathbf{N}^e = \begin{bmatrix} N_1^e & 0 & N_2^e & 0 & N_3^e & 0 & N_4^e & 0 \\ 0 & N_1^e & 0 & N_2^e & 0 & N_3^e & 0 & N_4^e \end{bmatrix} \tag{3.8}$$

and

$$\mathbf{u}^e = \begin{bmatrix} u_{r1} & u_{z1} & u_{r2} & u_{z2} & u_{r3} & u_{z3} & u_{r4} & u_{z4} \end{bmatrix}^T \tag{3.9}$$

from the displacements found, strains can be obtained using the following relations:

$$e_{rr} = \frac{\partial u_r}{\partial r} \qquad e_{zz} = \frac{\partial u_z}{\partial z} \qquad e_{\theta\theta} = \frac{u_r}{r} \qquad 2e_{rz} = \frac{\partial u_r}{\partial z} + \frac{\partial u_z}{\partial r} \tag{3.10}$$

Alternatively, strain-displacement relations can be shown in a matrix form:

$$\mathbf{e} = \begin{bmatrix} e_{rr} \\ e_{zz} \\ e_{\theta\theta} \\ 2e_{rz} \end{bmatrix} = \mathbf{B}\,\mathbf{u}^e \tag{3.11}$$

Where, matrix $\mathbf{B}$ is obviously composed of gradients of the shape functions.

$$\mathbf{B} = \begin{bmatrix} \frac{\partial N_1^e}{\partial r} & 0 & \frac{\partial N_2^e}{\partial r} & 0 & \frac{\partial N_3^e}{\partial r} & 0 & \frac{\partial N_4^e}{\partial r} & 0 \\ 0 & \frac{\partial N_1^e}{\partial z} & 0 & \frac{\partial N_2^e}{\partial z} & 0 & \frac{\partial N_3^e}{\partial z} & 0 & \frac{\partial N_4^e}{\partial z} \\ \frac{N_1^e}{r} & 0 & \frac{N_2^e}{r} & 0 & \frac{N_3^e}{r} & 0 & \frac{N_4^e}{r} & 0 \\ \frac{\partial N_1^e}{\partial z} & \frac{\partial N_1^e}{\partial r} & \frac{\partial N_2^e}{\partial z} & \frac{\partial N_2^e}{\partial r} & \frac{\partial N_3^e}{\partial z} & \frac{\partial N_3^e}{\partial r} & \frac{\partial N_4^e}{\partial z} & \frac{\partial N_4^e}{\partial r} \end{bmatrix} \tag{3.12}$$

Manipulation of the weak form gives us the following equation (Reddy, 1993[28]):

$$\mathbf{0} = \mathbf{K}^e\mathbf{u}^e - \mathbf{f}^e \tag{3.13}$$

In the above equation, $\mathbf{K}^e$ is the element stiffness matrix and $\mathbf{f}^e$ is the element force vector. They can be expressed mathematically as:

$$\mathbf{K}^e = \int_{\Omega^e} r\mathbf{B}^T\mathbf{E}\mathbf{B}\,\mathrm{d}\Omega \tag{3.14}$$

and

$$\mathbf{f}^e = \int_{\Gamma^e} r\mathbf{N}^T\hat{\mathbf{t}}\,\mathrm{d}\Gamma \tag{3.15}$$

In the equation 3.14, **E** is called the stiffness matrix and it contains information about material properties. In this study, isotropic material is assumed, therefore **E** matrix can be formed as follows:

$$\mathbf{E} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\nu \end{bmatrix} \tag{3.16}$$

In the above equation, E is the Young's modulus and ν is the Poisson's ratio. Please note the difference between the material stiffness matrix **E** and the constant E. For the purpose of this study, $\nu = 0.3$ is assumed in the following case studies.

As it is discussed in the section 3.4, Jacobian matrix is needed in order to convert integration into summation. For this purpose, Jacobian matrix is defined as follows:,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial r}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial r}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1^e}{\partial \xi} & \frac{\partial N_2^e}{\partial \xi} & \frac{\partial N_3^e}{\partial \xi} & \frac{\partial N_4^e}{\partial \xi} \\ \frac{\partial N_1^e}{\partial \eta} & \frac{\partial N_2^e}{\partial \eta} & \frac{\partial N_3^e}{\partial \eta} & \frac{\partial N_4^e}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{3.17}$$

In the case of quadrilateral elements above equation3.17 becomes:

$$\mathbf{J} = \frac{1}{4} \begin{bmatrix} \eta-1 & 1-\eta & 1+\eta & -\eta-1 \\ \xi-1 & -\xi-1 & 1+\xi & 1-\xi \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{3.18}$$

Determinant of the Jacobian matrix ($|\mathbf{J}|$) is simply,

$$|\mathbf{J}| = J_{11}J_{22} - J_{21}J_{12} \tag{3.19}$$

3.3 Mesh generation

Meshing of the design domain determines the size of the elements, therefore it determines the accuracy of the analysis and affects the topology of the optimized structure. Hence, in this study finer meshes are used as much as the computational power allowed. Generally, the total number of elements in the design domain is chosen to be between 5000-7000 in this study.

As it is the case for most topology optimization practices, the cross section of the design domain is assumed to be rectangular. Therefore the mesh is generated in the form of a solid rectangle. It is divided into uniform smaller squares forming each element. An example of this meshing is given in the below figure 3.2 with 4400 elements.

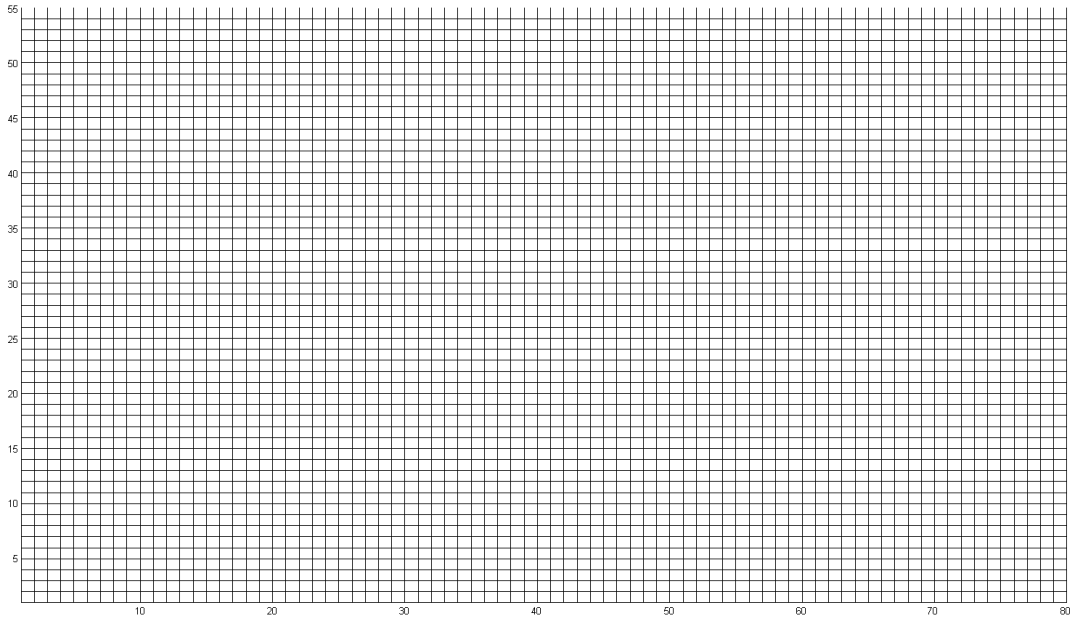

Figure 3.2: An example of discretization of the design domain by 55×80 mesh

Even though the mesh is uniform in the cross sectional plane; that does not mean each element is the same in three-dimensional space. Due to the differences in radial positions of the ring elements they have different volumes. Therefore, even if the mesh is considered to be uniform; volume differences between the elements should be taken into account while determining sensitivity numbers or formulating element removal-addition processes.

BESO has the advantage of using the same mesh throughout the whole optimization process. That is, initially formed mesh remains unchanged after each element addition-removal operation. Keeping the same mesh throughout the process, results in rougher boundaries due to the relatively coarse initial mesh. However it saves computational effort of generating mesh in each iteration and at the end of the optimization process it still gives idea about the optimal topology. Additionally, it should be noted that ultimate goal of the topology optimization is not to obtain a well-defined structure but to obtain a rough understanding of the optimal topology of the structure.

3.4 Integration Rule

Stiffness matrix of an element is found in the form of an integration as follows:

$$K^e = \int_{\Omega^e} rB^{\mathrm{T}} EB \, \mathrm{d}\Omega \tag{3.20}$$

and the force vector is found as follows:

$$f^e = \int_{\Gamma^e} rN^{\mathrm{T}} \hat{t} \, \mathrm{d}\Gamma \tag{3.21}$$

where Ω denotes the element domain, Γ denotes the element boundaries and $\hat{t}$ is the traction vector. For these integrations, an efficient numerical integration method needs to be chosen. Because linear shape functions are assumed in the element formulation, 2 point Gauss quadrature integration method is sufficient. This method gives exact result for polynomials up to order of 3, therefore it is quite an efficient method for the needs of this study. 2 point Gauss quadrature can be summed up as follows:

$$\int_{-1}^{1} f(x) \, \mathrm{d}x = c_1 f(x_1) + c_2 f(x_2) \tag{3.22}$$

where

$$c_1 = 1, \quad c_2 = -1, \quad x_1 = \frac{-1}{\sqrt{3}}, \quad x_2 = \frac{1}{\sqrt{3}} \tag{3.23}$$

for one dimensional case. For the integration of the force vector, one dimensional Gauss rule is used on the element boundary that the traction is applied. On the other hand, for the integration of the stiffness vector, 2 dimensional Gauss rule should be used on the element domain. In that case, weights and integration points change as follows:

$$c_1 = c_2 = c_3 = c_4 = 1 \tag{3.24}$$

$$x_1 = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right), \quad x_2 = \left(\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right), \quad x_3 = \left(\frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right), \quad x_4 = \left(\frac{-1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right) \tag{3.25}$$

After the Gauss quadrature rule is applied, the stiffness matrix integration becomes a simple summation in the form of:

$$K^e = \sum_{k=1}^{2} \sum_{l=1}^{2} w_k \, w_l \, B^{\mathrm{T}}(\xi_k, \eta_l) \, E \, B(\xi_k, \eta_l) \, r(\xi_k, \eta_l) \, J_\Omega(\xi_k, \eta_l) \tag{3.26}$$

and the force vector integration becomes:

$$f^e = \sum_{k=1}^{2} w_k \, N^{\mathrm{T}}(\xi_k, \eta_k) \, \hat{t}(\xi_k, \eta_k) \, r(\xi_k, \eta_k) \, J_\Omega(\xi_k, \eta_k) \tag{3.27}$$

In the equations 3.26 and 3.27, upper limits of summations are 2 because 2-point Gauss rule is applied. In these equations, w denotes the weight factor, ξ and η denote the integration points and the J_Ω is the determinant of the Jacobian at the integration points. In equation 3.26, two indices indicate four summation points that is, (ξ_k, η_l) has four different values. on the other hand, equation 3.27 has one index because there are only 2 integration points that is, (ξ_k, η_k) has two different values.

In finite element analysis, singularities may emerge when the **B** matrix is evaluated near the symmetry axis (r becomes close to zero) due to axisymmetric formulation of the **B** matrix. In order to prevent this problem, one should not choose the Gaussian integration points near the symmetry axis (Clayton and Rencis 1999)[10]. In this study, this problem is avoided by choosing the Gaussian points at the inner region of the element, therefore integration points can never be on the symmetry axis.

CHAPTER 4

CASE STUDIES

In the previous sections, an optimization methodology is presented. Algorithm of this methodology and mathematical formulations are given in detail. In order to validate presented methodologies, two different structural optimization problems from literature are selected to be solved. First one of these problems is the dome optimization in zero gravity taken from the study of Cherkaev and Palais (1996)[7]. Secondly the turbine disk optimization problem is solved and compared to the solution found in the study of Liu et al. (2005)[25]. One additional problem is prepared in accordance with Prof. Dr. Suha Oral; and solution to this problem is presented. Details of these problems are explained in the following chapters.

4.1 Problem 1

In order to evaluate the performance of the presented BESO method , firstly the dome optimization problem is solved. This problem and its solution is taken from the study of Cherkaev and Palais (1996)[7]. This problem is basically the optimization problem of an axisymmetric plate. It is a long studied problem, by researchers like Cheng and Olhoff (1981)[6] and Kirsch (1989)[21]. The study of Cherkaev and Palais in some sense decides this two decade long discussion.

This problem has a square shaped design domain in 2D. One side of this domain is coincident with the axis of symmetry ($r = 0$) and other side coincident with the line $r = 1$. That is, cross section of the design domain is 1 unit in length

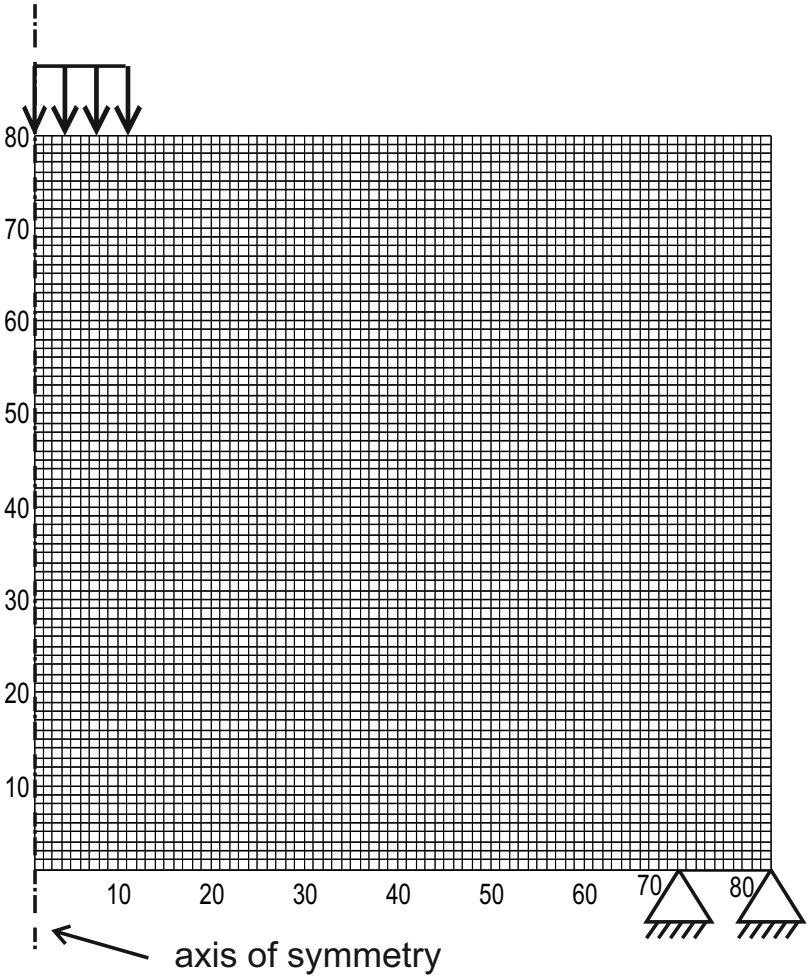

Figure 4.1: Graphical illustration of boundary conditions on the meshed domain

and 1 unit in height. Applied force is a distributed force acting vertically at the upper boundary of the design domain. Applied force has the intensity of 0.02 units and it acts on the nodes between the lines $r = 0$ and $r = 0.125$. On the other hand, support is at the lower boundary of the design domain. The lowermost nodes between the lines $r = 0.875$ and $r = 1$ are fixed in both vertical direction and horizontal direction. In the figure 4.1, boundary conditions are shown graphically on the 2-dimensional finite element mesh. Young's modulus is assumed as 1000 and Poisson's ratio is assumed as 0.3. As stated In earlier chapters, in this study BESO starts from the full design; that is , all the elements are solid initially.

For the solution of this problem, some parameters should be decided on in order

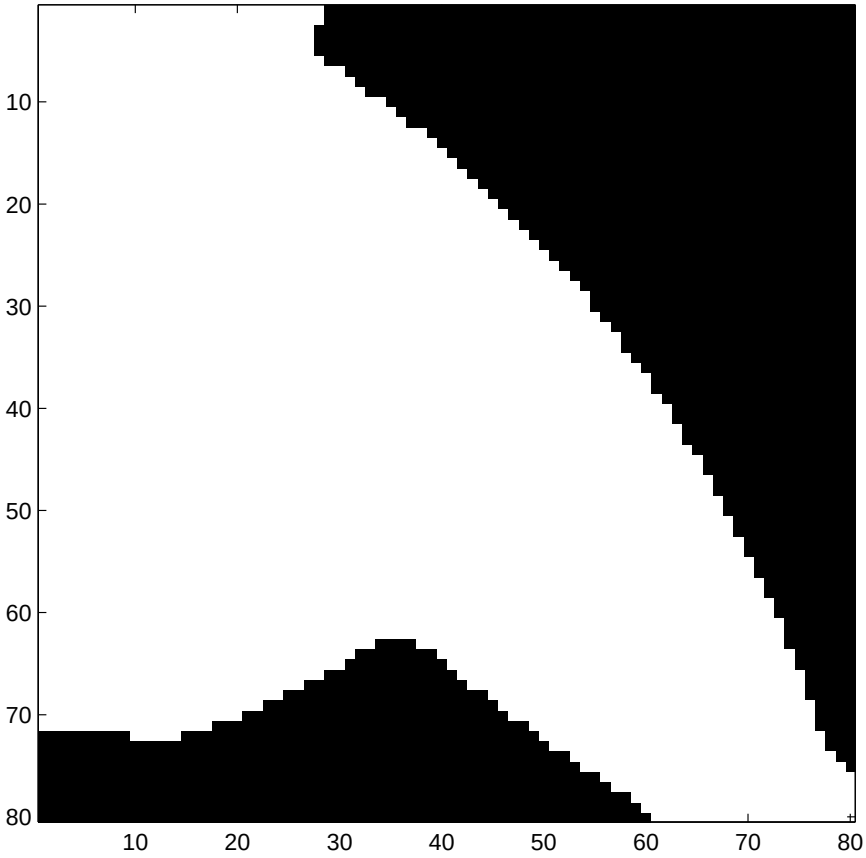

Figure 4.2: Evolving structure at volume fraction of 50%

to have the best performance out of the optimization process. These parameters are; evolutionary ratio (ER), convergence criterion (τ), maximum addition ratio (AR_{max}), final volume ratio (V^*) and sensitivity number filtering radius (r_{min}). Chosen values of those parameters are as follows:

$$\begin{aligned} ER = 0.01 \quad r_{min} = 0.05 \quad AR_{max} = 0.05 \\ V^* = 0.2 \qquad \tau = 10^{-4} \end{aligned} \tag{4.1}$$

With these chosen parameters, optimization process has been started with 80×80 mesh (i.e. with 6400 elements). Algorithm has started with removing the upper right elements as expected, because these elements have lowest energy levels and they have the highest volumes. An intermediate step is given in the figure 4.2 where volume fraction is 0.50. The elements that are removed in early steps can be seen as black in this figure.

At the end of the process, optimized structure is found as in the figure 4.3:

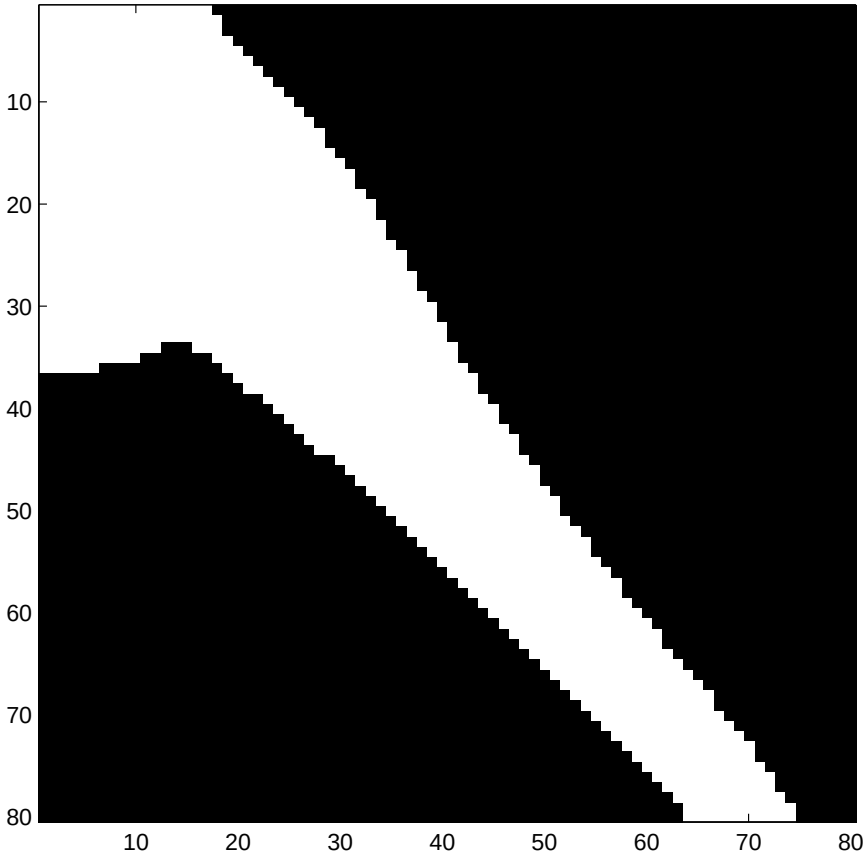

Figure 4.3: Optimized structure at final volume ratio of 0.2

Results of the optimization shows that the optimum structure has a cone-like shape in three-dimensions. It is thicker at the region where the traction is applied; and it is thinner at the radially outer regions because of larger volume of outer elements.

This result can be considered as the stiffest structure topology to carry the given load with given support conditions. However this solution is specific to the previously defined, final volume fraction V^*. Obviously, this topology would differ if another volume fraction was specified to the final design.

Evolution of the objective function can be seen in the figure 4.4. It can be seen that it has a fairly smooth trend. It increases up to around 80^{th} iteration because volume fraction constantly decreases. Around the 80^{th} step, volume fraction reaches V^* and it stays constant. Beyond this point, only a slight decrease in the objective function is seen, however this does not lead to a dramatical change in topology. This phase would continue longer if a smaller τ was chosen. In this

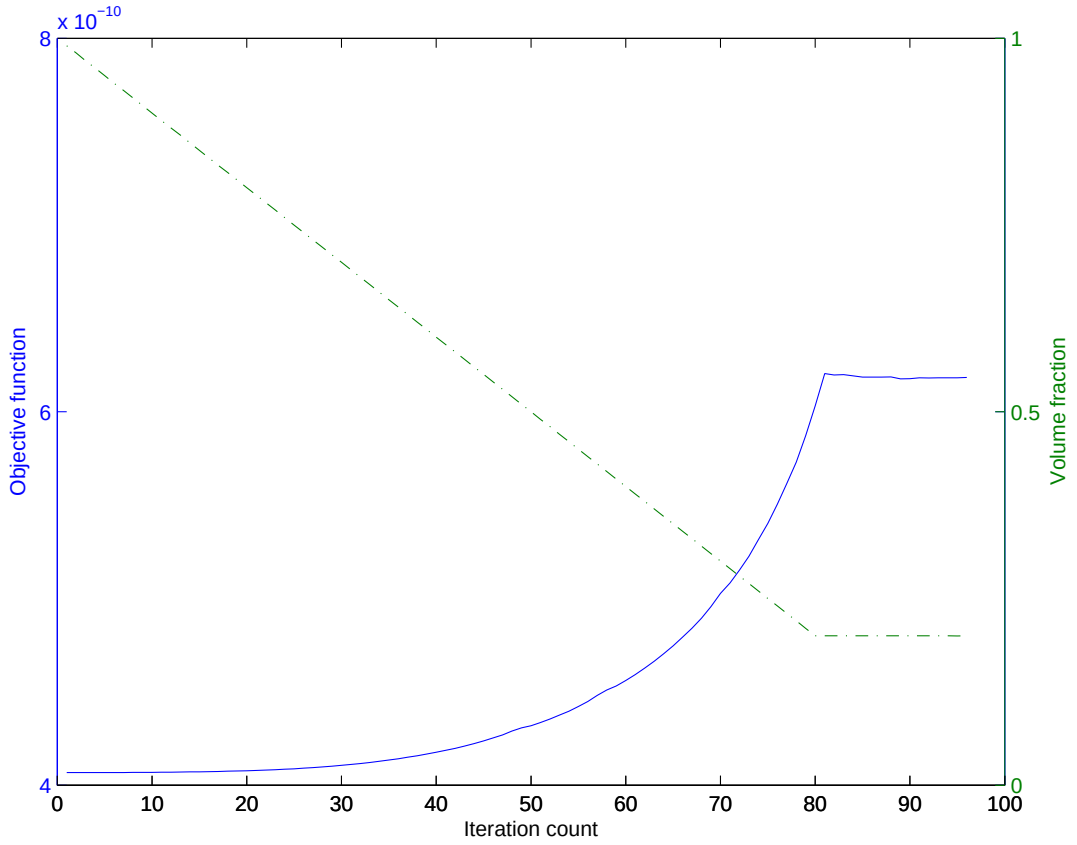

Figure 4.4: Evolution and convergence of the objective function

case, it took up to 97^{th} iteration for the structure to converge to an optimum.

When the solution (figure 4.3) is compared to the solution of Cherkaev and Palais (figure 4.5), a similar topology is observed. However, much coarser mesh in the study of Cherkaev and Palais is obvious. Thanks to the finer mesh in figure 4.3, boundary definitions and details like rounded corners are observed much better. Another difference between these solutions is the use of gray elements (elements with intermediate mechanical properties). In the BESO method there is no gray element assumption, therefore each element is either solid or void. This leads to the difference in interpretation of the optimum structures in figure 4.3 and figure 4.5.

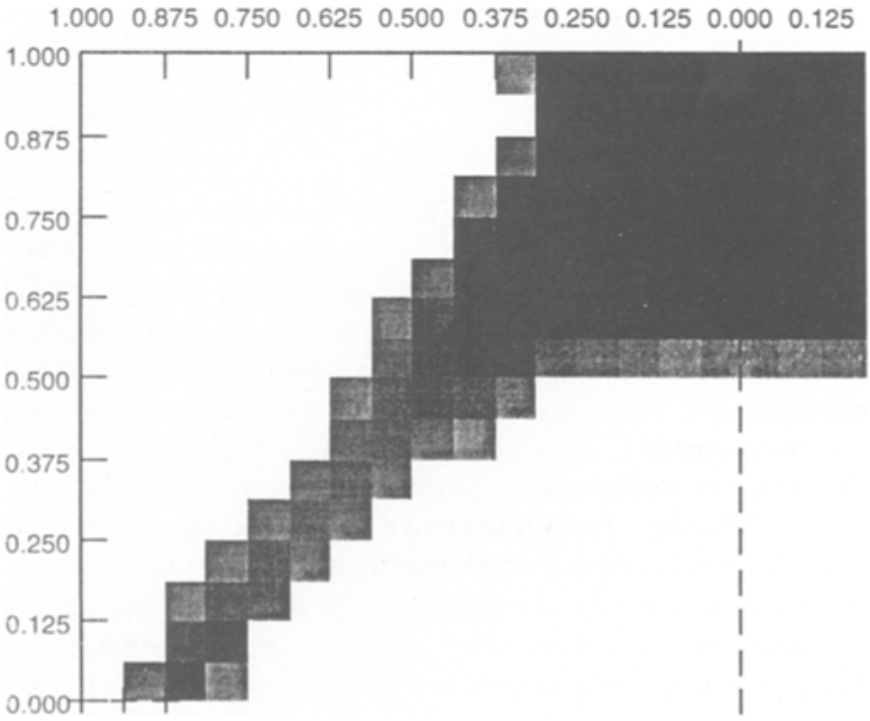

Figure 4.5: Results found by Cherkaev and Palais [7]

4.2 Problem 2

For the performance evaluation of the presented optimization method, the second problem to be solved is the optimization problem of a turbine disk. This problem has an importance of being a practical industrial design problem. Because of being used in many critical applications, turbine disk optimization have been studied for a long time. Considering the study of Donath (1912)[11], this problem is more than one century old. Some of the following studies on this topic are the ones of Stodola(1927)[36], Bhavikatti and Ramakrishnan, (1980)[5], Luchi et al.(1980)[26], Cheu(1989)[8] and Cheu (1990)[9]. In 2005, Liu et al.[25] has addressed this problem by applying Metamorphic Development method. They have considered this problem as a shape/topology optimization problem and they have adapted their method accordingly. The manipulations made on the BESO method are discussed later.

This problem has one axis of symmetry and one plane of symmetry. It is axisymmetric around the z-axis and it is considered to be symmetric about the r-axis. These symmetry conditions can be seen in detail in the figure

4.6. Because of axisymmetry, structure is modeled in two-dimensions with axisymmetric elements. In addition, because of r-symmetry, only the upper half of the disk is modeled and lowermost nodes are constrained in the z-direction. Finite element model of structure is in rectangular shape with left side is coincident with $r = 20$ line and right side is coincident with $r = 120$ line; lower boundary is on $z = 0$ line and upper boundary is on $z = 40$ line. Therefore it is 100mm wide and 40mm high. In their work; Liu et al. have considered three different loading conditions of the turbine disk. Those are thermal loads, body forces and blade loadings. For simplicity, only the blade loading case is taken from their study. Therefore the only loading is the distributed load acting outwards on the right hand side boundary ($p = 200$MPa).

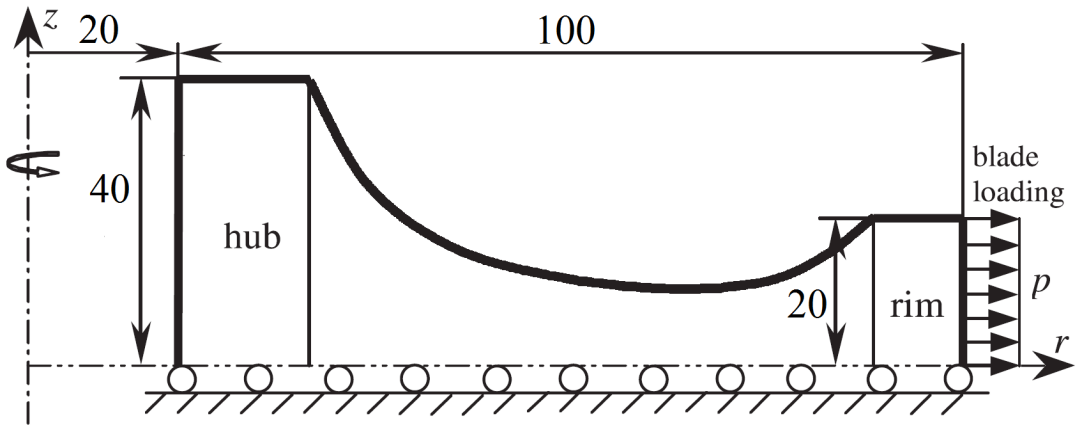

Figure 4.6: Boundary conditions of the optimization problem[25]

There are also constraints of the design domain that is, not all the elements are allowed to be removed. Firstly, the elements laying at the left of the lines $z = 115 - 3r$ and $z = 59 - r$ are excluded from the element addition-removal operations. Additionally, the elements that are to the right of $r = 115$ and below the $z = 20$ lines are excluded from the element addition-removal. All elements are initiated as solid elements, therefore constrained elements stay as solid ones during the whole optimization process. Those elements are also excluded from the volume fraction calculations, that is, the volume fraction term is valid only for the non-constrained elements. Mentioned constraints on the design domain can be seen from the figure 4.7.

As stated above, the turbine disk optimization is considered like a shape

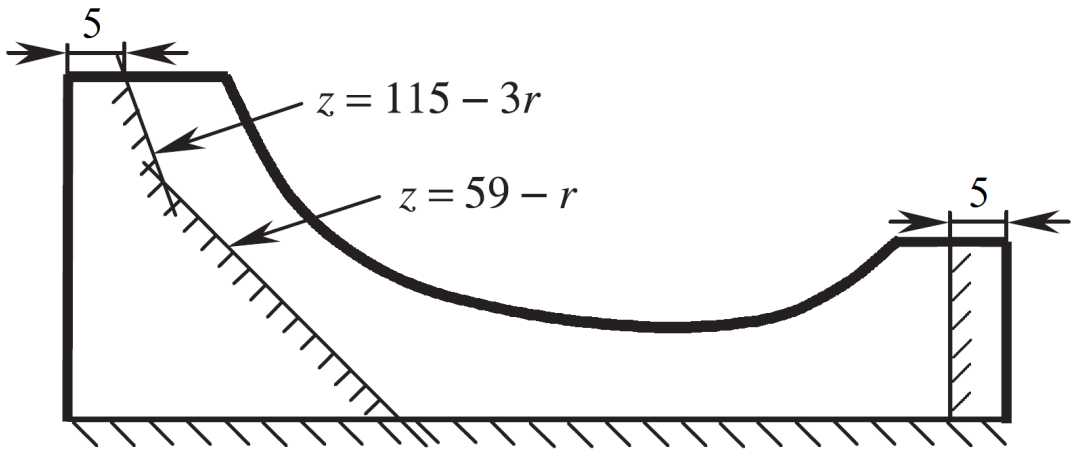

Figure 4.7: Constraints on the design domain of the problem[25]

optimization problem by Liu et al.. For the same reason, BESO method was manipulated by excluding the elements below the $z = 5$ line from the addition–removal operations. In this way, cavities occurring inside of the turbine disk are suppressed. As it can be seen from the results (figure4.9), this constraint does not interfere with the surface of the disk. Therefore it does not affect the shape of the disk.

In order to determine which side the elements belong, element center points are taken as reference. If an element center lies exactly on the constraint line, it is taken as a constrained element.

In this problem, a 48×120 mesh (i.e., 5760 elements) is used for discretization of the analysis domain. Young's modulus is taken as $180.36GPa$ and Poisson's ratio is assumed as 0.3. Parameters of optimization are taken as follows:

$$\begin{aligned} ER = 0.01 \quad r_{min} = 10 \qquad AR_{max} = 0.05 \\ V^* = 0.1 \qquad \tau = 10^{-3} \end{aligned} \tag{4.2}$$

Optimization process has been started with the set-up mentioned above. Upper-right elements are removed in the earlier steps. Element removal gradually reached the constrained elements and curvature becomes visible at the mid-section of the disk structure (figure 4.8). Final structure can be seen from the figure 4.9.

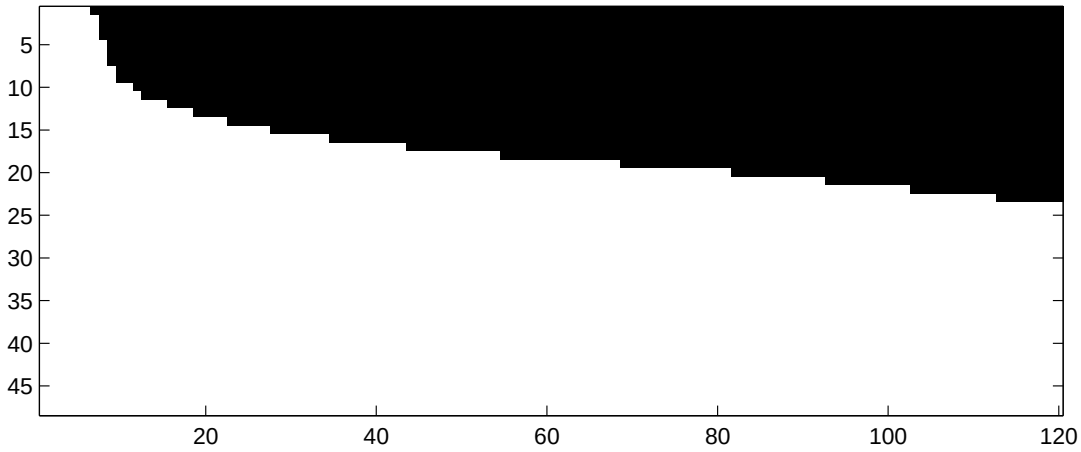

Figure 4.8: Evolving structure at volume fraction of 50%

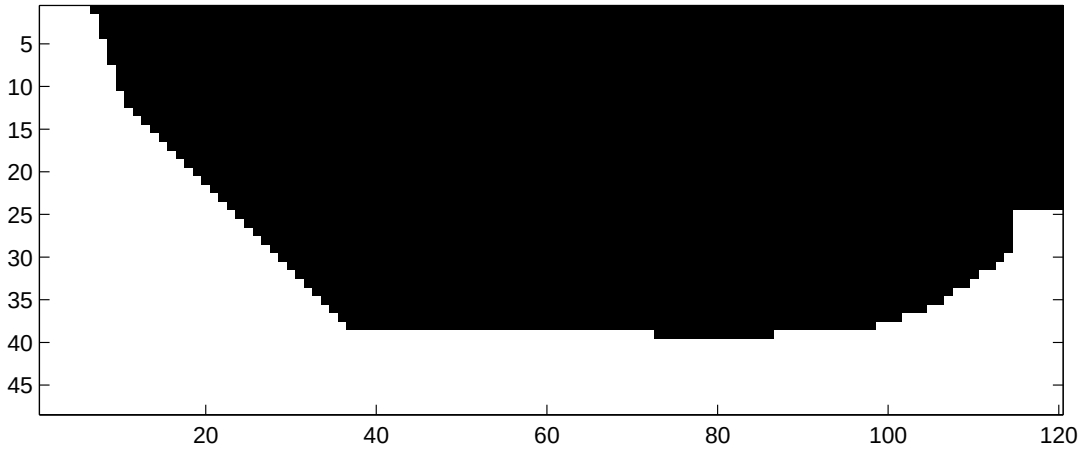

Figure 4.9: Optimum structure of the turbine disk found by BESO method

Results show that optimum disk has a constant thickness mid-section and a thin flange at the region of application of distributed loads. Joining these two, there is a fillet; possibly because of stress concentrations and resulting high strain energies. Topology of the inner region is completely defined by the constraints therefore one cannot talk about the effect of optimization procedure in this region. It should be noted that this resulting optimum structure is obtained at volume fraction of 0.10, which means 10% of the volume has been used when the constrained elements are excluded. If the volume fraction is calculated by including all elements (constrained and non-constrained ones) the final volume fraction is found as 30%. However this value is not used in the optimization process as a parameter.

When the evolution of objective function is investigated, a smooth trend is

observed. the slope of the objective function gets steeper as the evolution proceeds. This means more stiffness is sacrificed in the later steps, for the same amount of volume removal. Reason to this is, higher strained elements are left towards the end of the process. After the volume fraction reaches a steady state at around 90^{th}, convergence is obtained relatively quickly, in around 10 iterations. This is highly dependent on the pre-selected value of τ. If a smaller τ was selected, convergence criterion would get tighter and more iterations would be needed until convergence is obtained. The whole optimization process concludes in 102 iterations. Related figure can be seen below (figure4.10)

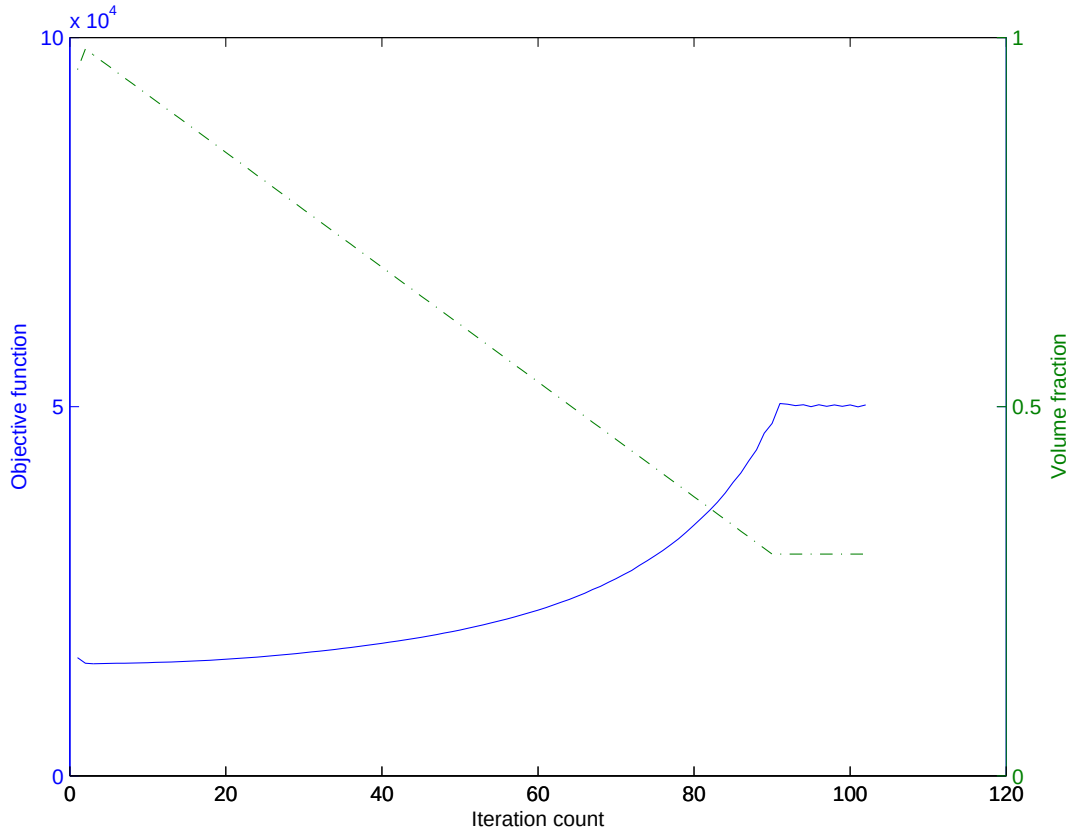

Figure 4.10: Evolution of the objective function and volume fraction

When the results are compared to that of Lui at al.[25], a close resemblance is observed. However actual sizes are different because of the difference in the methods used. In addition, boundary definitions are different in two results. In figure 4.11, boundaries are fairly smooth, because boundary element method is used as the analysis method. On the other hand in figure 4.9, rough boundaries are observed. Reason to that is the shape of quadrilateral elements. As the

mesh resolution increases, the boundary definitions improve however the basic topology stays the same.

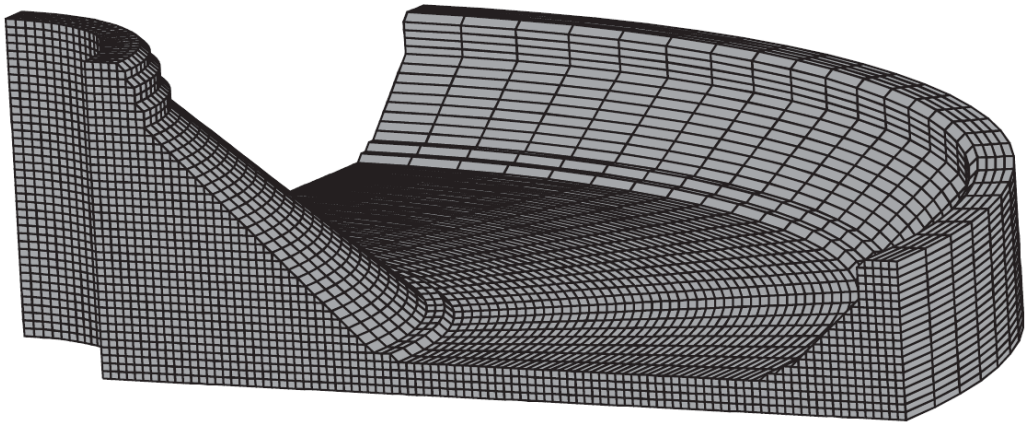

Figure 4.11: Optimum structure of the turbine disk found by Liu et al.[25]

4.3 Problem 3

Third problem in this study is the optimization of a thick walled cylinder. In this problem, design domain is assumed to be in rectangular shape. It represents a hollow cylinder with thick walls therefore design domain is offset from the symmetry axis. Inner diameter is assumed to be 20mm and outer diameter is 50mm. Therefore design domain is 15mm wide and 60mm high while its inner boundary is 10mm offset from the z-axis. Distributed force is assumed to be applied from the inner boundary, therefore it is shown to be acting from the left hand side of the design domain in two-dimensional mesh. Distributed force is decreasing linearly in the increasing z-direction. It decreases to half of its value at the uppermost point, namely, it has the value $10N/mm$ at $z = 0$ and has $5N/mm$ at $z = 60$. The values in between are linear. This problem is symmetric only around the z-axis (axisymmetry). It has no symmetry about r-axis because load distribution is assumed to be variable, which breaks symmetry condition about r-axis.

In the problem definition there are no displacement constraints. However this not applicable in finite element formulation. Without any constraints, a rigid

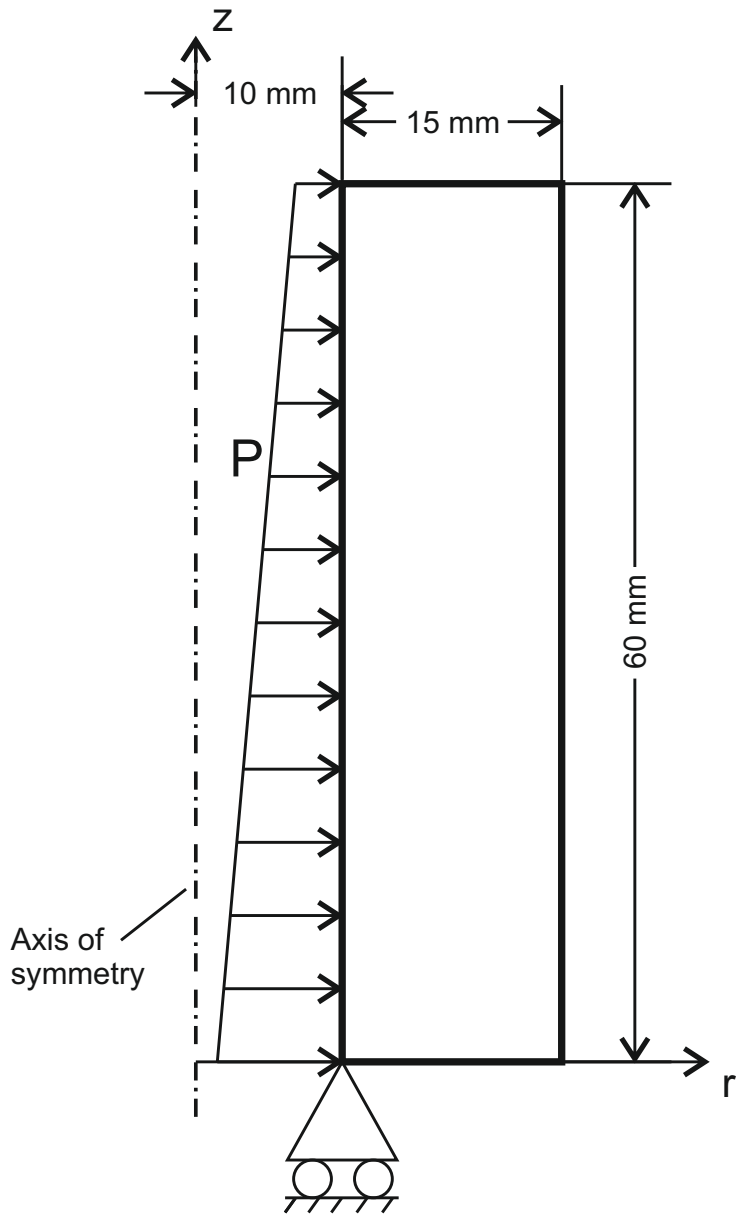

Figure 4.12: Graphical illustration of boundary conditions on the design domain

body motion is introduced in the z-direction. This redundant degree of freedom makes it impossible to find a solution. In order to avoid this problem, a z-direction constraint is applied at the lower left corner of the design domain. With the help of this constraint, structure is prevented from moving freely in space. This constraint could be applied at any point of the domain, the only reason is to eliminate the rigid body motion in z-direction. Dimensions and conditions defining the problem can be found in the figure 4.12.

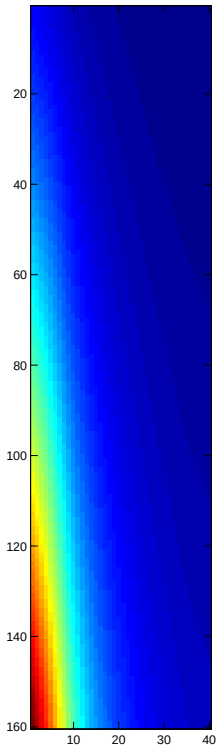

Figure 4.13: Analysis of fully solid design domain

As stated before, solution of this problem is not compared to that of another study from the literature. At this point, based on the previous analytical work of Timoshenko and Goodier (1951)[39] and Liang et al.(2008)[24], stress distribution can be expected to be higher at the inner boundary. Due to the proportionality of strain energy and stresses (equation 4.3), inner elements have higher strain energies and sensitivity numbers. This expectation is supported by the finite element analysis of the solid design domain (figure 4.13).

$$W = \frac{1}{2}\sigma : \epsilon \tag{4.3}$$

In this problem, a mesh of 160×40 is used therefore elements are square shaped in two dimensional mesh. In total there are 6400 elements and they are uniform throughout the design domain. In this case, Young's modulus is taken as $1000MPa$ and Poisson's ratio as 0.3. The parameters of the optimization

method can be given as below:

$$\begin{aligned} ER = 0.01 \quad r_{min} = 3 \qquad AR_{max} = 0.05 \\ V^* = 0.3 \qquad \tau = 10^{-4} \end{aligned} \tag{4.4}$$

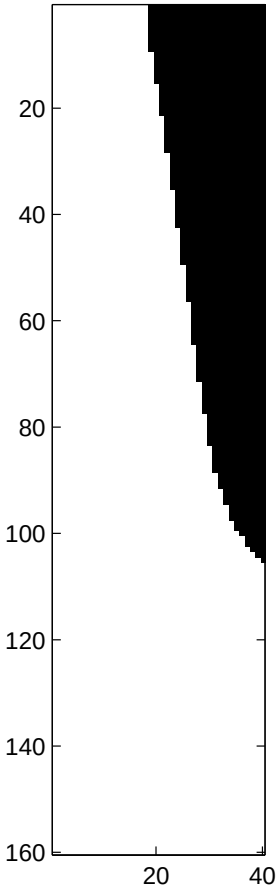

Figure 4.14: Evolving structure at volume fraction of 70%

When the optimization is started from the full design, element removal begins form the upper-right corner as expected. The outer elements are strained less therefore they have smaller sensitivity numbers. In addition, at the upper regions the load is less and upper elements are strained less. Therefore upper right elements are removed first and structure evolves near inner boundary as expected (figure 4.14).

Resulting optimum structure can be seen in the figure 4.15. It is seen that optimum structure is concentrated near the inner boundary and it gets gradually thinner as the load decreases in z-direction. This result agrees with expectations

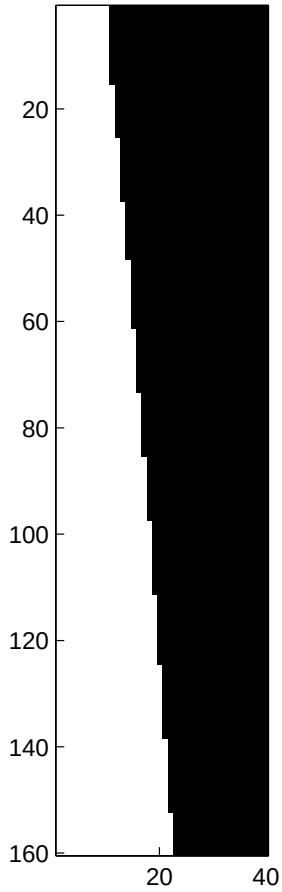

Figure 4.15: Optimum structure of the finite length thick walled cylinder

based on analytical results, as stated earlier.

Optimization process has taken 84 iterations in total. Evolution of the volume fraction is completed in 70 iterations. In this phase evolution trend of the objective function is very smooth without any discontinuities or jumps. This is because the evolving topology stays as a one-piece solid topology through the process and material removal at the outer boundaries resulted in a gradual increase in mean compliance. After V^* is reached, topology has converged to the optimum in 14 iterations, again without any oscillations in objective function. Evolution of the objective function and the volume fraction of the structure can be found in the figure 4.16.

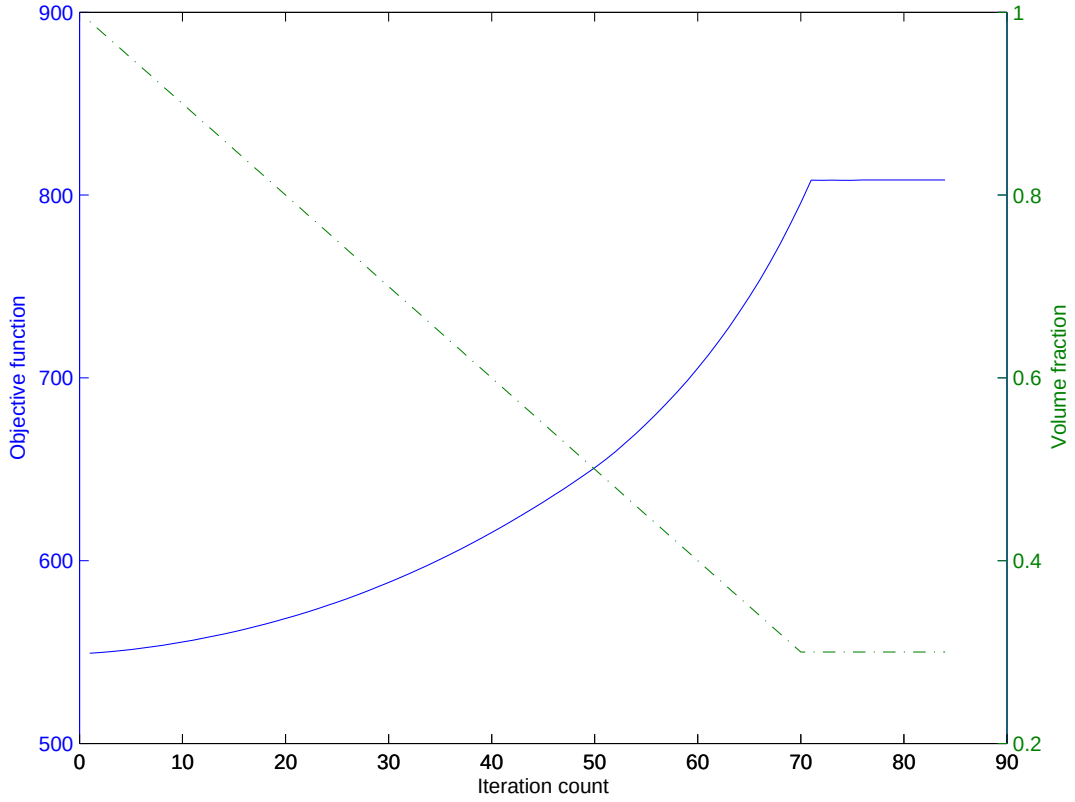

Figure 4.16: Evolution of the objective function and volume fraction

CHAPTER 5

SUMMARY AND CONCLUSION

In this chapter, summary of this study and the work done is presented. Following that, this study is concluded and final remarks are given. Lastly, recommendations are made for the possible future studies in this area.

5.1 Summary

This study is started by defining the problem and mentioning important studies from the literature. At this point focus was mostly kept on the optimization procedure. Historical development of BESO is explained and some further improvements on this method are mentioned.

In the chapter2, the outline of the methodology is given. BESO methodology is explained in detail and mentioned improvements are added to it. Total algorithm is presented clearly in figure 2.1. Steps of this algorithm are presented in separate sections while mentioning case problems solved later. At the end, reader has a complete picture of the optimization method used in this study.

In chapter 3, the details about the analysis procedure are given. Reasons of selecting specific methods are explained and mathematical backgrounds lying underneath are presented. A satisfactory explanation was aimed to be given in this section. However, giving a complete explanation about the finite element method is beyond the scope of this study. Therefore, in this section the focus is kept on reasoning the selected parameters and methods while mentioning mathematical roots without detailed derivations.

Up to this point, a complete methodology was built. Lastly, in chapter 4, performance of this methodology is evaluated by applying three different case problems. Two of these problems were chosen from the literature while one scenario was created specially for this study. Solutions to these problems are found and two of solutions are compared to those from literature.

This complete optimization methodology is applied through MATLAB code. Code is modified and it is different for each case problem, however methodology is common for each case. MATLAB code used in the solution of turbine disk problem is given in appendix A as the representative code.

5.2 Conclusion

In this study, new BESO method investigated and applied to various problems. From the solution of these problems, it is concluded that BESO method is an effective and efficient optimization method. It is used with volume ratios around 30%. However, optimization process can sometimes get unstable with lower volume fractions. Stability is also dependent on other optimization parameters such as evolutionary ratio, sensitivity averaging weights and sensitivity filtering radius. Therefore, extra attention should be given to optimization parameters in order to have the best out of the BESO method. Apart from some stability issues, BESO performs quite well. It has found the optimum topologies quite accurately. It is different than some of the optimization methods in a sense that it is an easy method to understand and implement. Therefore it is possible to modify it according to the needs of a specific problem. In this study, a finite element code is embedded into the optimization code and both of them are written specifically for this study. However, yet another advantage of the BESO is that it is easy to combine with commercial finite element programs. It does not require regeneration of the mesh, only the change of the element properties would be sufficient for element removal. Therefore it would be quite straightforward even if it is used with commercial FEM programs. BESO method is an intuitive way while having quite strong theoretical background, which makes it a powerful method. To conclude, the new BESO method is

even more effective with the recent developments and it sets a benchmark for structural topology optimization methods.

5.3 Recommendations for Future Work

For the future studies and upcoming studies, some recommendations can be given. The researcher who are planning to work with BESO method can focus their studies on the following topics:

- Although the focus is on compliance minimization in most studies, in practical applications stress constraints may be needed to be implemented into topology optimization process. Some researchers have implemented stress constraints into optimization methods like SIMP (Duysinx and Bendsøe, 1998[12] and Pereira et al., 2004[27]) and Simulated Annealing (Shim and Manoochehri, 1997[33]). In the future studies, researchers can focus on applying a similar stress constraint formulations to BESO methodology.
- Checkerboard problem can be addressed by using higher order finite elements therefore preventing discontinuities between element boundaries. Therefore, there will be no need for a separate subroutine against checkerboard problem.
- Multi-objective optimization may be applied using two or more objective functions.
- The optimization code can be manipulated accordingly and a GUI can be designed for the code in order to have a user friendly and flexible optimization tool.

REFERENCES

[1] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393, 2004.

[2] M. P. Bendsøe and N. Kikuchi. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71(2):197 – 224, 1988.

[3] M. P. Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1(4):193–202, 1989.

[4] M. P. Bendsøe and O. Sigmund. *Topology Optimization Theory, Methods and Applications*. Springer-Verlag Berlin, 2003.

[5] S. Bhavikatti and C. Ramakrishnan. Optimum shape design of rotating disks. *Computers & Structures*, 11(5):397–401, 1980.

[6] K.-T. Cheng and N. Olhoff. An investigation concerning optimal design of solid elastic plates. *International Journal of Solids and Structures*, 17(3):305 – 323, 1981.

[7] A. Cherkaev and R. Palais. Optimal design of three-dimensional axisymmetric elastic structures. *Structural Optimization*, 12(1):35–45, 1996.

[8] T.-C. Cheu. Sensitivity analysis and shape optimization of axisymmetric structures. *International Journal for Numerical Methods in Engineering*, 28(1):95–108, 1989.

[9] T.-C. Cheu. Procedures for shape optimization of gas turbine disks. In *ASME 1990 International Gas Turbine and Aeroengine Congress and Exposition*, pages V005T14A018–V005T14A018. American Society of Mechanical Engineers, 1990.

[10] J. D. Clayton and J. J. Rencis. Numerical integration in the axisymmetric finite element formulation. *Advances in engineering software*, 31(2):137–141, 2000.

[11] M. Donath. Die berechnung rotierender scheiben und ringe. *Berlin, Julius Springer*, 1912.

[12] P. Duysinx and M. P. Bendsøe. Topology optimization of continuum structures with local stress constraints. *International journal for numerical methods in engineering*, 43(8):1453–1478, 1998.

[13] C. A. Felippa. Introduction to finite element methods. *Course Notes, Department of Aerospace Engineeing Sciences, University of Colorado at Boulder, available at http://www. colorado. edu/engineering/Aerospace/CAS/courses. d/IFEM. d*, 2004.

[14] E. Hinton and J. Sienz. Fully stressed topological design of structures using an evolutionary procedure. *Engineering Computations*, 12(3):229–244, 1995.

[15] E. Holmberg. Stress and fatigue constrained topology optimization. 2013.

[16] X. Huang and Y. Xie. Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method. *Finite Elements in Analysis and Design*, 43(14):1039 – 1049, 2007.

[17] X. Huang and Y. Xie. *Evolutionary topology optimization of continuum structures : methods and applications*. John Wiley & Sons, Ltd., 2010.

[18] C. Jog. Topology design of structres subjected to periodic loading. *Journal of Sound and Vibration*, 253(3):687 – 709, 2002.

[19] C. S. Jog and R. B. Haber. Stability of finite element models for distributed-parameter optimization and topology design. *Computer Methods in Applied Mechanics and Engineering*, 130(3-4):203–226, 1996.

[20] F. Jouve. Structural shape and topology optimization. In G. Rozvany and T. Lewiński, editors, *Topology Optimization in Structural and Continuum Mechanics*. Springer, 2014.

[21] U. Kirsch. Optimal Topologies of Structures. *Applied Mechanics Reviews*, 42(8):223, 1989.

[22] L. Krog, A. Tucker, and G. Rollema. Application of topology, sizing and shape optimization methods to optimal design of aircraft components. In *Proc. 3rd Altair UK HyperWorks Users Conference*, 2002.

[23] Q. Li, G. Steven, and Y. Xie. A simple checkerboard suppression algorithm for evolutionary structural optimization. *Structural and Multidisciplinary Optimization*, 22(3):230–239, 2001.

[24] Y. Liang, H. Wang, and X. Ren. Analytical solution for spatially axisymmetric problem of thick-walled cylinder subjected to different linearly varying pressures along the axis and uniform pressures at two ends. *Science in China, Series G: Physics, Mechanics and Astronomy*, 51(1):98–104, 2008.

[25] J. S. Liu, G. T. Parks, and P. J. Clarkson. Topology/shape optimisation of axisymmetric continuum structures - A metamorphic development approach. *Structural and Multidisciplinary Optimization*, 29(1):73–83, 2005.

[26] M. Luchi, A. Poggialini, and F. Persiani. An interactive optimization procedure applied to the design of gas turbine discs. *Computers & Structures*, 11(6):629–637, 1980.

[27] J. Pereira, E. Fancello, and C. Barcellos. Topology optimization of continuum structures with material failure constraints. *Structural and Multidisciplinary Optimization*, 26(1-2):50–66, 2004.

[28] J. N. Reddy. *An introduction to the finite element method*, volume 2. McGraw-Hill New York, 1993.

[29] A. Rietz. Sufficiency of a finite exponent in simp (power law) methods. *Structural and Multidisciplinary Optimization*, 21(2):159–163, 2001.

[30] G. Rozvany. A critical review of established methods of structural topology optimization. *Structural and Multidisciplinary Optimization*, 37(3):217–237, 2009.

[31] G. Rozvany, O. Querin, Z. Gaspar, and V. Pomezanski. Extended optimality in topology design. *Structural and Multidisciplinary Optimization*, 24(3):257–261, 2002.

[32] J. A. Sethian and A. Wiegmann. Structural boundary design via level set and immersed interface methods. *Journal of computational physics*, 163(2):489–528, 2000.

[33] P. Y. Shim and S. Manoochehri. Generating optimal configurations in structural design using simulated annealing. *International journal for numerical methods in engineering*, 40(6):1053–1069, 1997.

[34] O. Sigmund. On the design of compliant mechanisms using topology optimization. *Mechanics of Structures and Machines*, 25(4):493–524, 1997.

[35] O. Sigmund and J. Petersson. Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 16(1):68–75, 1998.

[36] A. Stodola. Steam and gas turbines. vol. i. *IIMcGraw-Hill, New York*, pages 1039–1073, 1927.

[37] C. Talischi, G. H. Paulino, A. Pereira, and I. F. M. Menezes. Polygonal finite elements for topology optimization: A unifying

paradigm. *International Journal for Numerical Methods in Engineering*, 82(6):671–698, 2010.

[38] P. Tanskanen. The evolutionary structural optimization method: theoretical aspects. *Computer Methods in Applied Mechanics and Engineering*, 191(47–48):5485 – 5498, 2002.

[39] S. Timoshenko and J. Goodier. *Theory of Elasticity*. McGraw-Hill Book Company Inc., 1951.

[40] M. Y. Wang, X. Wang, and D. Guo. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1):227–246, 2003.

[41] X. Wang, M. Wang, and D. Guo. Structural shape and topology optimization in a level-set-based framework of region representation. *Structural and Multidisciplinary Optimization*, 27(1-2):1–19, 2004.

[42] Y. Xie and G. Steven. A simple evolutionary procedure for structural optimization. *Computers & Structures*, 49(5):885 – 896, 1993.

[43] Y. Xie and G. Steven. *Evolutionary structural optimization*. Springer-Verlag London, 1 edition, 1997.

[44] G. S. X.Y. Yang, Y.M. Xei and O. Querin. Bidirectional evolutionary method for stiffness optimization. 37:1483–1488, 1999.

[45] M. Zhou and G. Rozvany. The coc algorithm, part ii: topological, geometrical and generalized shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 89(1):309–336, 1991.

[46] M. Zhou and G. Rozvany. On the validity of eso type methods in topology optimization. *Structural and Multidisciplinary Optimization*, 21(1):80–83, 2001.

APPENDIX A

MATLAB CODE USED IN THE PROBLEM 2

The m-scripts used as the optimization and analysis code are given in the following pages.

A.1 Main Function `ESO.m`

```
% --------------------ESO----------------------

clc
clear all
close all
imgpath='C:\Users\Oguz\Dropbox\tez\fortran-dislin\axis
symmetric\matlab-'...
    'turbine disk - true'; % path of the images to be saved

%----------STEP 1 DEFINE FE MESH
n=6 %6 max
xel=20*n; yel=8*n;
left=ones(yel,xel*3/20);
middle=ones(yel,xel*15/20);
right=vertcat(zeros(yel/2,xel*2/20),ones(yel/2,xel*2/20));
solid=horzcat(left,middle,right);

%----------STEP 2 DEFINE BESO PARAMETERS
```

```
tau=0.0001% convergence criterion
error=1; %only before the loops begin
ARmax=0.05; %maximum element addition ratio
ER=0.01;%evolutionary ratio
N=5; % error < tau in the last N steps
Vstar=0.1;%final volume fraction
V=1; %this depends on the initial config
flag=0;
k=0;

while ( (error > tau) || ((V-ER) >= Vstar) )
k=k+1;
%----------STEP 3 FEA
[C(k), sens_num_mat, sens_num_arr, Vi, element_pos,felems]
=analysis(solid);

%----------STEP 4 SENS AVERAGING
if (flag==0)
    sens=sens_num_arr;
    flag=1;
end
sens=sens.*0.5+sens_num_arr.*0.5;

%----------STEP 5 CALC TARGET VOLUME
V=max(V-ER,Vstar);

%----------STEP 6 ADD-DELETE
[solid,solid_arr]=addel(V,ARmax,sens,solid,Vi, element_pos,
felems);

%----------STEP 6 plot
figure(1)
colormap(gray)
```

```
imagesc((solid)) %pcolor draws upside down
pbaspect([20,8,1]) % set the asepct ratio of the plot
drawnow

if (abs(V-0.5)<0.001)
    saveas(gcf,fullfile(imgpath,'turbine_midstep'), 'epsc')
end

%----------STEP 6 error calculation
if (V-Vstar < 0.001)
error= abs( sum(C(k-(1:N)+1)) - sum(C(k-N-(1:N)+1)) )
/(sum(C(k-(1:N)+1)));
end
Vhist(k)=sum(Vi(:).*solid_arr(:))./sum(Vi(:));
Vhist(k)
V
C(k)
end

saveas(gcf,fullfile(imgpath,'turbine_son'),'epsc')

figure(2)
x=1:k;
[ax,Cline,Vhistline]=plotyy(x,C,x,Vhist);
set(Cline,'LineStyle',':');
set(Vhistline,'LineStyle','-');
ylabel(ax(1),'Objective function') % label left y-axis
ylabel(ax(2),'Volume fraction') % label right y-axis
xlabel(ax(2),'Iteration count') % label x-axis

saveas(gcf,fullfile(imgpath,'turbine_objective'), 'epsc')

disp('FIN')
```

A.2 Function addel.m

```
function [ solid_mat , solid_all] = addel( Vfrac, ARmax, sens,
...
    solid_mat,Vi, element_pos,felems)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here


% clc
% clear all
% close all
% load('C:\Users\Oguz\Desktop\matlab.mat')
% Vfrac=0.5;
% ARmax=0.1;
% solid_mat=[  0 1 1 1 0    ; 1 1 0 1 1     ; 1 1 1 1 1  ];
% sens=     [ 18;13;19;15;3 ; 8;19;17;15;13 ; 3;6;1;14;8 ];
% Vi=       [ 2;3;4;5;6     ; 2;3;4;5;6     ; 2;3;4;5;6  ];


%solid matrix is turned into array
[yel,xel]=size(solid_mat);
solid_all=[];
for i=yel:-1:1
    solid_all=horzcat(solid_all,solid_mat(i,:));
end
[~,elnum]=size (solid_all);
element_numbers(:,1)=1:elnum;

%calculate constraints
const1(:,1)=3*element_pos(:,1)+element_pos(:,2)-115;
const2(:,1)=element_pos(:,1)+element_pos(:,2)-59;
```

```
%find the constrained elements
j=1;
for i=1:elnum
    if ((const1(i)<=0)||(const2(i)<=0)  || (element_pos(i,2)<5)
|| ...,
            (element_pos(i,1) >115 && element_pos(i,2)<20))
        const_elem(j,1)=i;
        j=j+1;
    end
end
const_elem=unique(const_elem);

element_numbers(const_elem)=[];
sens(const_elem)=[];
solid=solid_all; solid(const_elem)=[];
Vi(const_elem)=[];


[~,elnum]=size (solid);
% sort senstitivity numbers and element numbers accordingly
[~, index] = sort(sens,'descend');
element_numbers=element_numbers(index);
solid = solid(index);
Vi= Vi(index);
total_vol=sum(Vi);
V=total_vol*Vfrac;
%-----step 1
%calculate treshold sens number
for i=1:elnum
    sumVi(i)=sum(Vi(1:i));
end
[~,th_el_seq] = min(abs(sumVi-V));
```

```
void_seq=find(~solid);
added_elems_seq=void_seq(find(void_seq <= th_el_seq));
added_volume=sum(Vi(added_elems_seq));
AR=added_volume/total_vol;

%if armax is not violated, make solid-void transformations
according to
%sensitivity number order
if (AR<ARmax)
    solid(:)=0;
    solid(1:th_el_seq)=1;
%     solid_const_unsorted(element_numbers(:))=solid(:);
end



%if ARmax is violated first calculate elements to be added, then
remove
%elements to satisfy volume constraint.
if (AR>ARmax)

    %calculate; up to which void element the addition will be
performed.
    added_volume=total_vol*ARmax; %volume to be added according
to ARmax
    void_Vi=Vi(void_seq);
    [~,voidnum]=size(void_seq);

    void_sumVi=0;
    for i=1:voidnum
        void_sumVi=void_sumVi+void_Vi(i);
        if (void_sumVi >= added_volume)
```

```
            voidtosolid=i-1;
            break
        end
    end
    solid(void_seq(1:voidtosolid))=1;

    % calculate the volume to be removed.
    removed_volume=sum(solid.*transpose(Vi))-V;
    [~,solidnum]=size(find(solid));
    solid_seq=find(solid);
    solid_Vi=Vi(solid_seq);
    solid_sumVi=0;

    for i=solidnum:-1:1
        solid_sumVi=solid_sumVi+solid_Vi(i);
        if (solid_sumVi >= removed_volume)
            solidtovoid=i+1;
            break
        end
    end
    solid(solid_seq(solidtovoid:solidnum))=0;

%     solid_const_unsorted(element_numbers(:))=solid(:);

end
solid_all(element_numbers(:))=solid(:);
solid_temp=solid_all;
for i=yel:-1:1
solid_mat(i,:)=solid_temp(1:xel);
solid_temp(1:xel)=[];
end

end
```

A.3 Function `analysis.m`

```
function [ mean_comp , sens_matrix_fil, sens_array_fil, Vi,
element_pos,...
    felems ] = analysis( Solid_old )
% UNTITLED Summary of this function goes here
%   Detailed explanation goes here
% clc
% clear all
% close all

% n=1
% xel=20*n; yel=8*n;
% Solid_old=ones(yel,xel) ;
% Solid_old(1:4,4:20)=0;


E=180.36*1000;
nu=0.3;
xmin=1e-10;

Solid_old(Solid_old==0)=xmin; %zeros are changed to xmin, in
order to
% prevent unconstrained dofs


% ---------I-I-I-INPUT---------------%
[yel,xel]=size(Solid_old);
a=20; b=120; d=40; p=10;
felems=[xel:xel:(xel*yel/2) ];  % force applied at elements: f
mat
%integration i da duzenle
fxdnodes=[ 2:2:((xel+1)*2) ]; %fixed degree of freedoms
```

```
trac=[p ; 0]; % this can be a 2 dim matrix as elem num increases
theta=1; % angle of revolution of the axisymmetric cross section
% (full rev.: 2*pi)
rmin=10;  %sensitivity number filtering radius
% ---------I-I-I-INPUT---------------%

% ---------solid void info is changed into array form
solid=[];
for i=yel:-1:1
    solid=horzcat(solid,Solid_old(i,:));
end
% ---------solid void info is changed into array form

elnum=yel*xel;
sens_array=zeros(elnum,1);
dx=(b-a)/xel; dy=d/yel;
xnode=xel+1; ynode=yel+1;
nodenum=xnode*ynode;
freenodes=1:2*nodenum; freenodes(fxdnodes)=[];%non-fixed degree
of freedoms
Disp=zeros(2*nodenum,1); disp=zeros(8,elnum);
Kmat=zeros(2*nodenum); Fmat=zeros(2*nodenum,1); % global f and k
matrices
fmat=zeros(8,elnum); % elemnt force vector
elem_nodecoor=zeros(4,2,elnum);

cmat=(E/((1+nu)*(1-2*nu)))*[1-nu, nu, nu, 0; nu, 1-nu, nu, 0; nu,
nu, ...
    1-nu, 0; 0, 0, 0, (1/2-nu)];
ksi=[-1/sqrt(3) 1/sqrt(3)];
eta=[-1/sqrt(3) 1/sqrt(3)];

%element-node connectivities and
```

```
%element number-node coordinates informations are calculated
for j=1:yel
    for i=1:xel
        % nodes are numbered in ccw direction
        % numbering starts from bottom left, x-direction first,
y-dir later
        elem_nodecoor(:,:,((j-1)*xel)+i)=[
((i-1)*dx+a),((j-1)*dy) ; ...
            (i*dx+a),((j-1)*dy) ; (i*dx+a),(j*dy) ;
((i-1)*dx+a),(j*dy) ];
        % in conn() rows contain node numbers for the element
that has the
        %same number as that  row
        % conn (element number, node numbers)
        conn(((j-1)*xel)+i,:)=[(j-1)*xnode+i, (j-1)*xnode+i+1,
...
            j*xnode+i+1, j*xnode+i];
    end
end

%node number- node coordinates info
for i=1:nodenum
    [row,col]=find(conn==i);
    nodecoor(i,:)=elem_nodecoor(col(1),:,row(1));
end

for i=1:elnum
    element_pos(i,:)= mean(elem_nodecoor(:,:,i));
end
for i=1:elnum
    Vi(i,1)= element_pos(i,1)*theta;
end
```

```
% ------------------------------STIFFNESS MAT
CALC-----------------------
for i=1:elnum
    % kmat * 2 pi eklenecek
    kmat1=theta.*transpose(bmat(ksi(1),  eta(1),
elem_nodecoor(:, :, i))) ...
        *cmat*bmat(ksi(1), eta(1),  elem_nodecoor(:, :,
i)).*det(Jcb(ksi(1),  ...
        eta(1), elem_nodecoor(:, :, i))).*rpos(ksi(1), eta(1),
...
        elem_nodecoor(:, :, i));
    kmat2=theta.*transpose(bmat(ksi(1),  eta(2), elem_nodecoor(:,
:, i))) ...
        *cmat*bmat(ksi(1), eta(2),  elem_nodecoor(:, :,
i)).*det(Jcb(ksi(1), ...
        eta(2), elem_nodecoor(:, :, i))).*rpos(ksi(1) , eta(2),
...
        elem_nodecoor(:, :, i));
    kmat3=theta.*transpose(bmat(ksi(2), eta(1), elem_nodecoor(:,
:, i))) ...
        *cmat*bmat(ksi(2), eta(1), elem_nodecoor(:, :,
i)).*det(Jcb(ksi(2), ...
        eta(1), elem_nodecoor(:, :, i))).*rpos(ksi(2), eta(1),
...
        elem_nodecoor(:, :, i));
    kmat4=theta.*transpose(bmat(ksi(2), eta(2), elem_nodecoor(:,
:, i))) ...
        *cmat*bmat(ksi(2), eta(2), elem_nodecoor(:, :,
i)).*det(Jcb(ksi(2), ...
        eta(2), elem_nodecoor(:, :, i))).*rpos(ksi(2), eta(2),
...
        elem_nodecoor(:, :, i));
    kmat(:,:,i)=kmat1+kmat2+kmat3+kmat4;
```

```
    kmat(:,:,i)=kmat(:,:,i).*solid(i);
end
% ------------------------------STIFNESS MAT
CALC-----------------------


% ------------------------------TRACTION FORCE MAT
CALC-------------------
%--------------------set the boundary(ksi (x) or eta (y)) of
applied force
%----------------------------also, the direction from the
jacobian. i.e.
%----------------------------2,2 or 1,1
for i=1:size(transpose(felems))
    fjcb1=Jcb(1,eta(1),elem_nodecoor(:,:,felems(i)));
    frpos1=rpos(1,eta(1),elem_nodecoor(:,:,felems(i)));
    fjcb2=Jcb(1,eta(2),elem_nodecoor(:,:,felems(i)));
    frpos2=rpos(1,eta(2),elem_nodecoor(:,:,felems(i)));
    %!!!!! fmat theta ile carpilacak mi? (angle of revolution)
    fmat(:,felems(i)) =transpose(Nmat(1,eta(1))) *trac
.*fjcb1(1,1) .*frpos1;
    fmat(:,felems(i)) =transpose(Nmat(1,eta(2))) *trac
.*fjcb2(1,1)...
        .*frpos2+fmat(:,felems(i));
end
% ------------------------------TRACTION FORCE MAT
CALC------------------


% --------------------SCATTERING----------------------
for i=1:elnum
    map_seq=[ 2*conn(i,1)-1, 2*conn(i,1), 2*conn(i,2)-1,
2*conn(i,2), ...
```

```
        2*conn(i,3)-1, 2*conn(i,3), 2*conn(i,4)-1, 2*conn(i,4) ];
    for j=1:8
        for k=1:8
            Kmat(map_seq(j),map_seq(k)) =kmat(j,k,i)
+Kmat(map_seq(j),...
                map_seq(k));
        end
        Fmat(map_seq(j)) =fmat(j,i) +Fmat(map_seq(j));
    end
end
% ----------------------SCATTERING-----------------------


% ------------------SOLVING-----------------------
kfree=Kmat; kfree(fxdnodes,:)=[];kfree(:,fxdnodes)=[];
ffree=Fmat; ffree(fxdnodes)=[];
ufree=kfree\ffree;
Disp(freenodes)=ufree;
% ---------------------SOLVING-------------------------

% -------------------------------SENS NUMBER
CALC-------------------------
for i=1:elnum
    map_seq=[ 2*conn(i,1)-1, 2*conn(i,1), 2*conn(i,2)-1,
2*conn(i,2),...
        2*conn(i,3)-1, 2*conn(i,3), 2*conn(i,4)-1, 2*conn(i,4) ];
    disp(:,i)=Disp(map_seq);

    sens_array(i) =(1/2*transpose(disp(:,i)) *kmat(:,:,i)
*disp(:,i))./Vi(i);
end
% -----------------------------SENS NUMBER
CALC-------------------------
```

```
% ------------------------------SENS NUMBER
FILTERING--------------------

%finding nodal sensitivity numbers
nodal_sens_array=zeros(nodenum,1);
for i=1:nodenum
    [connected_elems, ~ ]= find(conn==i);
    [connected_elem_num,~]=size(connected_elems);
    w=1/connected_elem_num;
    for j=1:connected_elem_num
        nodal_sens_array(i)=w*sens_array(connected_elems(j)) ...
            +nodal_sens_array(i);
    end
end

%finding filtered elemental sensitivity numbers


for i=1:elnum
    for j=1:nodenum
        w(j)=rmin-sqrt((nodecoor(j,1)-element_pos(i,1))^2+...
            (nodecoor(j,2)-element_pos(i,2))^2);
    end
    w(w<0)=0;
    sens_array_fil(i,1)=(w*nodal_sens_array)/sum(w);
end
% ------------------------------SENS NUMBER
FILTERING------------------

sens_temp=sens_array_fil; sens_matrix_fil=zeros(yel,xel);
for i=yel:-1:1
    sens_matrix_fil(i,:)=sens_temp(1:xel);
```

```
    sens_temp(1:xel)=[];
end

mean_comp=1/2*transpose(Fmat)*Disp;
end
```

A.4 Function bmat.m

```
function [ bmat ] = bmat(ksi, eta, nodecoor)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

J=Jcb(ksi,eta, nodecoor);
J11=J(1,1); J12=J(1,2); J21=J(2,1); J22=J(2,2);
detJ=J11*J22-J12*J21;
ngrad(1,:)=1/4*[eta-1, 0, 1-eta, 0, 1+eta, 0, -1-eta, 0];
ngrad(2,:)=1/4*[ksi-1, 0, -1-ksi, 0, 1+ksi, 0, 1-ksi, 0];
ngrad(3,:)=1/4*[0, eta-1, 0, 1-eta, 0, 1+eta, 0, -1-eta];
ngrad(4,:)=1/4*[0, ksi-1, 0, -1-ksi, 0, 1+ksi, 0, 1-ksi];

bmat=(1/(detJ)).*[J22, -J12, 0, 0; 0, 0, -J21, J11; 0, 0, 0, 0;
-J21,...
    J11, J22, -J12]*ngrad;

denom= (1-ksi)*(1-eta)*nodecoor(1,1)+ (1+ksi)* (1-eta)*
nodecoor(2,1)+ (1+ksi)...
    *(1+eta)*nodecoor(3,1)+ (1-ksi)*(1+eta)*nodecoor(4,1);
alp9=(1-ksi)*(1-eta)/denom;
alp10=(1+ksi)*(1-eta)/denom;
alp11=(1+ksi)*(1+eta)/denom;
alp12=(1-ksi)*(1+eta)/denom;
```

```
bmat(3,:)=[alp9, 0, alp10, 0, alp11, 0, alp12, 0];

end
```

A.5 Function Jcb.m

```
function [ jcb ]= Jcb(ksi, eta, nodecoor)

ngrad=[eta-1, 1-eta, 1+eta, -eta-1;ksi-1, -ksi-1, 1+ksi, 1-ksi];
jcb=1/4.*ngrad*nodecoor;

end
```

A.6 Function Nmat.m

```
function [ Nmat ] = Nmat( ksi, eta )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

N1=(1-ksi)*(1-eta)*1/4;
N2=(1+ksi)*(1-eta)*1/4;
N3=(1+ksi)*(1+eta)*1/4;
N4=(1-ksi)*(1+eta)*1/4;

Nmat=[N1, 0, N2, 0, N3, 0, N4, 0; 0, N1, 0, N2, 0, N3, 0, N4];

end
```

A.7 Function rpos.m

```
function [ rpos ] = rpos( ksi, eta, nodecoor )
```

```
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

N1=(1-ksi)*(1-eta)*1/4;
N2=(1+ksi)*(1-eta)*1/4;
N3=(1+ksi)*(1+eta)*1/4;
N4=(1-ksi)*(1+eta)*1/4;

rpos=nodecoor(1,1)*N1+nodecoor(2,1)*N2+ nodecoor(3,1)*N3+
nodecoor(4,1)*N4;
end
```