Shape and Topology Synthesis of Structures using a Sequential Optimization Algorithm

by

Ashok V. Kumar

M.S., Mechanical Engineering
The University of Michigan
June, 1990

B.Tech., Mechanical Engineering
Indian Institute of Technology
June, 1988

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy
in Mechanical Engineering

at the

Massachusetts Institute of Technology

September, 1993

Signature of Author _____
Department of Mechanical Engineering
August 18, 1993

Certified by _____
Professor David C. Gossard
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Department Committee

Shape and Topology Synthesis of Structures using a Sequential Optimization Algorithm

by

Ashok V. Kumar

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Mechanical Engineering.

ABSTRACT

In this thesis, a new method is proposed for the design optimization of structural components where both its shape and topology are optimized. The boundaries of the shape of the structure are represented using contours of a "shape density" function and the shape is optimized using a sequential approximate optimization algorithm. The advantage of this shape representation is that both shape and topology of the structure can be modified and optimized by the optimization algorithm. In this shape representation, regions where the density function value is higher than a threshold value are considered to be the interior of the shape. The contour of the density function corresponding to the threshold value is defined as the boundary of the shape. The shape density function is defined over a feasible domain and is represented by a continuous piece-wise interpolation over the finite elements used for structural analysis. The values of the density function at the nodes serve as the design variables of the optimization problem. The material properties of the structure are assumed to depend on the density function. Many approximate material property-density relations have been studied. The shape and topology of structural components are optimized with the objective of minimizing the compliance subject to a constraint on the total mass of the structure. The number of design variables in this optimization problem is very large and the functions are computationally expensive to evaluate. A sequential approximate optimization algorithm was developed that is well suited for this application. The method generates a sequence of sub-problems iteratively using a first order approximation for the objective function and sets bounds on the variables using the barrier method. This algorithm uses only the first order derivatives of the objective function and constraints and does not require excessive number of function evaluations. The shape representation and the optimization algorithm proposed here provide a computationally efficient method for designing optimal structural components.

Thesis Advisor:	Professor David C. Gossard
Thesis Committee:	Professor Mary C. Boyce
	Professor Mark J. Jakiela

Acknowledgments

I wish to express my sincere gratitude to my thesis advisor Professor David C. Gossard for his support, encouragement and council without which this thesis would not have been possible. The quality of this research has also greatly benefited from the inputs and suggestions from Professor Mary C. Boyce and Professor Mark J. Jakiela. I would like to thank them for their encouragement, help and constructive criticism. Many thanks are also due to Professor Richard Garrett for his interest in my work and his advice in furthering my career.

This research was supported by Ford Motor Company. Their financial support is sincerely appreciated. I would also like to express my appreciation to the Computer-Aided Design Laboratory at MIT for providing excellent technical facilities and an environment conducive to research.

My life at MIT was greatly enriched by friends at the CADLAB. I have greatly benefited from discussions with George Celnicker, Kenji Shimada, Barbara Balents, Lian Fang, David Wallace, Jayaraman Krishnasamy, Colin Chapman, Minho Chang and many other friends at MIT.

Finally, I would like to thank my parents for their love and encouragement and for instilling in me the drive and motivation that made this work possible.

Table of Contents

List of Figures

List of Tables

Nomenclature

Shape representation

ϕ	Shape density function
ϕ_i	Value of shape density function at the ith node
ϕ_{th}	Threshold value corresponding to shape boundary
L_i, i=1,2,3	Area coordinates or linear interpolation functions for triangles
Δ	Area of each triangular finite element
$L(\mathbf{u})$	Compliance of the structure
W	Weight / Volume of the structure
W_o	Maximum allowed weight for the structure
x_i, y_i	Nodal coordinates of node *i* in the finite element mesh

Finite element analysis

$C^m(\Omega)$	Function whose partial derivatives of order upto *m* are continuous in Ω.
$H^m(\Omega)$	Sobolev space of order *m*
$\mathbf{X}, \{X_i\}$	Spatial coordinates for the current (/modified) shape
$\mathbf{X}_o$	Spatial coordinates for the initial shape
Ω	Current domain of structural analysis (after shape change)
Ω_o	Initial domain of structural analysis
Ω_e	The domain of one element of the finite element mesh
Γ	Boundary of the structural domain
Γ_o	Part of the boundary on which displacement boundary conditions are specified
Γ_t	Part of the boundary on which external traction is acting
$\mathbf{t}$	Traction acting on the boundary Γ_t
$\mathbf{f}$	Body force acting on the structure
$\mathbf{u}, \{u_x, u_y\}$	Displacement vector field
$\sigma, \{\sigma\}$	Stress field
$\varepsilon, \{\varepsilon\}$	Strain field
$[\mathbf{D}]$	Elasticity matrix for plane stress - plane strain
$[\mathbf{D}_l]$	Elasticity matrix with linear approximate dependence on density function
$[\mathbf{D}_i]$	Elasticity matrix with i^{th} order dependence on density function

$\delta\mathbf{u}$	Virtual displacement vector field
$\delta\varepsilon$	Virtual strain field
n_a	Number of degrees of freedom in the finite element analysis model
$\mathbf{u}_h$	Discrete representation of the displacement field
$\{\varepsilon^e\}$	Strain within a constant strain element 'e'.
$\{u_h^e\}$	Displacements associated with nodes of the element 'e'.
[B]	Coefficient matrix for strain-displacement relation for constrain strain triangular finite elements
$\mathbf{K}$, [K]	Global stiffness matrix
$[K_e]$	Element stiffness matrix
$\mathbf{F}$	Resultant external force vector
λ_a	Adjoint variables for sensitivity analysis
T	Transformation from the initial shape to the modified shape
$\mathbf{V}_i$	Design velocity with respect to the ith design variable
$\mathbf{K}^e$	Stiffness matrix or differential operator for composite material
$\mathbf{K}_H$	Homogenized stifness matrix for composite material
a,b	Void dimensions in the microstructure for homogenization method
θ	Microstructure orientation for homogenization method
E	Young's Modulus of Elasticity
ν	Poisson's ratio

Nonlinear programming

n	Number of design variables
m	Number of equality constraints
r	Number of inequality constraints
x_i, $\mathbf{x} \in R^n$	Design variables, vector of design variables
$\mathbf{x}^k$	Value of the design variables at the k_{th} iteration
$f(\mathbf{x})$	Objective function, $f:R^n \rightarrow R$
$\mathbf{h}(\mathbf{x})$	Equality constraint functions, $\mathbf{h}:R^n \rightarrow R^m$
$\mathbf{g}(\mathbf{x})$	Inequality constraint functions, $\mathbf{g}:R^n \rightarrow R^r$
$\{l_i\}$, $\mathbf{l}$	Lower bounds on the variables
$\{u_i\}$, $\mathbf{u}$	Upper bounds on the variables
$\{l_i^k\}$, $\mathbf{l}^k$	Lower move limits on the variables
$\{u_i^k\}$, $\mathbf{u}^k$	Upper move limits on the variables
$F_L(\mathbf{x})$	Linear approximation of a function $f(\mathbf{x})$
$F_I(\mathbf{y})$	Function $f(\mathbf{x})$ linearized with respect to intermediate variables $\mathbf{y}$

$F_R(\mathbf{x})$	Function f(**x**) linearized with respect to reciprocal variables
$F_C(\mathbf{x})$	Conservative approximation of a function f(**x**)

1 Introduction

1.1. Goals and Motivation

Shape and topology optimization involves the application of design optimization techniques to synthesize mechanical components so that a certain property of the component is optimized subject to some performance constraints. When either this property or the performance constraints relate to the structural properties of the component, it is referred to as structural optimization.

Structural optimization automates synthesis of mechanical components whose primary function is to provide structural support. As opposed to analysis, where a given design's performance is evaluated by simulating its mathematical model, synthesis involves searching among many feasible designs to arrive at one that best satisfies a certain design goal. Therefore, synthesis involves repeated analysis in search of better designs. Like most design problems, structural design involves many design goals and criteria (also referred to as the functional requirements). It is often impossible to account for all the design goals or criteria simultaneously. Just as one constructs a model for analysis by neglecting all but the most important phenomena, design optimization problems for synthesis should ideally consider only the most important design criteria. It is common practice to state the more critical criteria as design constraints and select one criteria as the design objective that is to be optimized.

The design process has been decomposed into various stages or phases [Shigley_83, Kirsch_81]. The commonly identified phases are: recognition of need and definition of problem, that is, the formulation of the functional requirements; conceptual design; design optimization; and detailing. In the context of structural design, the functional requirements could include the loads to be supported by the structure, the location and type of supports on the structure, a limit on the weight of the structure. etc. Conceptual design is usually associated with the creative part of the design where the designer uses his ingenuity and

engineering judgment to select the "type" of the design. For example, the overall topology of the structure, the type of the structure (truss, frame or plate etc.), the material etc. Design optimization, as stated earlier, involves repeated design analysis to find the optimal design among the many that satisfy the functional requirements. This optimization is often automated using computational techniques. Finally, in the detailing stage, details that relate to tolerance, manufacturing specifications, aesthetics, etc. are added. Figure 1.1 gives a schematic representation of the design process.

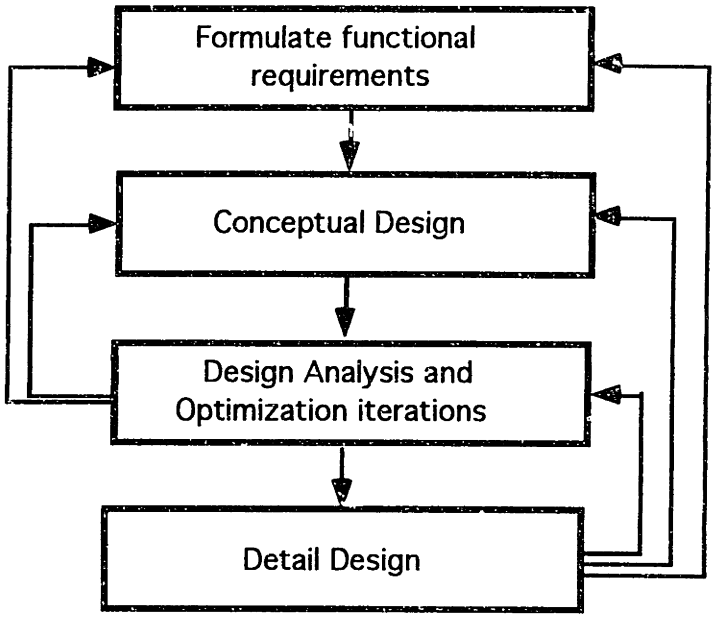


Figure 1.1. Stages of the design process

Design is iterative at each phase as illustrated in figure 1.1. At the conceptual stage many alternatives or design concepts are generated. Each such concept or design type may be expressed in terms of design parameters and the optimal values of these parameters may be found iteratively. In structural design, the topology generation for the structure is traditionally associated with the conceptual phase of design [Kirsch_81]. The designer may consider many alternative topologies iteratively. During the optimization stage, the topology is fixed and only certain parameters such as the cross-section of the elements of a truss or parameters describing the boundary of the geometry are varied. This type of approach may yield sub-optimal shapes due to the inability of the approach to modify the topology (or the layout) during the optimization. As a result, the optimal shape predicted by such optimization techniques depends on the initial guess. If the initial starting shape selected by the designer at the conceptual stage, does not have the optimal topology, the final shape computed by such an approach will be sub-optimal. The truly optimal shape may require

the creation of one or more new boundaries to change the topology of the component being designed. This is illustrated in figure 1.2.

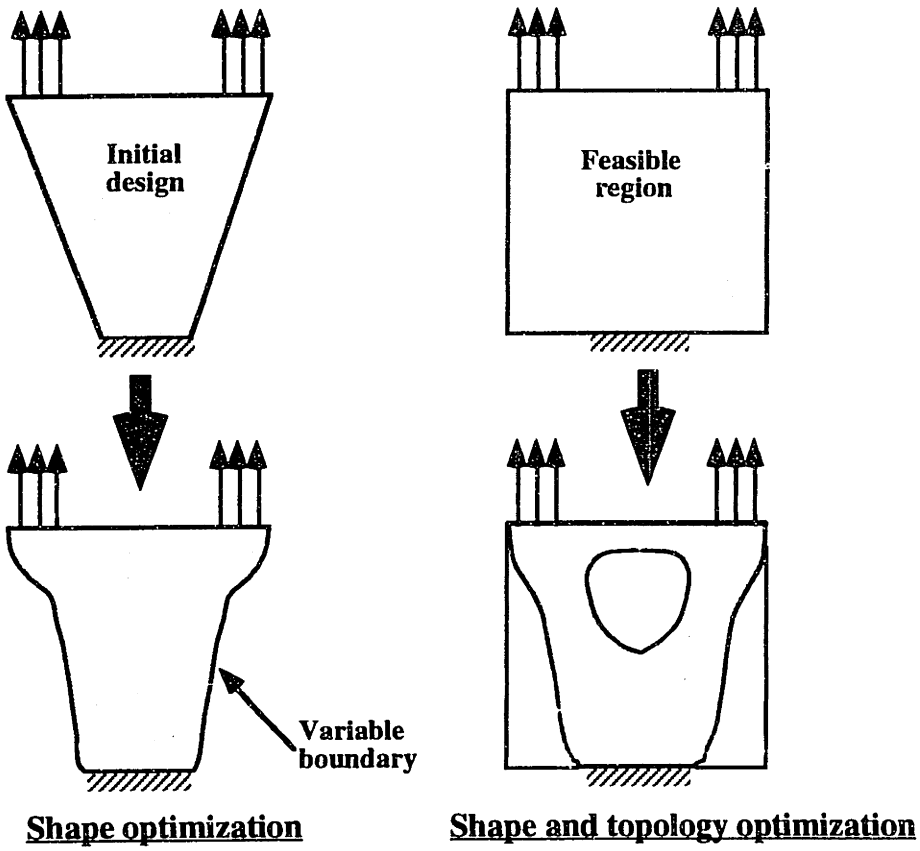


Figure 1.2. Shape versus topology optimization

Starting from an initial rectangular shape, the shape on the left in figure 1.2 is obtained by varying just two of the boundaries. While this shape has a better strength to weight ratio than the starting rectangular shape, even better results may be obtained by creating a new variable boundary. The shape on the right is obtained by adding a new boundary to create a hole, thereby removing some under-utilized material from the center of geometry to get an even better strength to weight ratio. Conventional techniques of shape optimization that use parametric curve or surface boundaries have difficulty in identifying regions where it is beneficial to create a new boundary and it is cumbersome to keep track of newly added boundaries and the associated variables.

Combined shape and topology optimization needs a more flexible geometry representation that allows the topology to change during the optimization. The principal advantage is that the designer then does not have to guess the optimal topology or specify the variable boundaries. Instead, the designer specifies a feasible region within which the

component has to fit as well as the support locations (boundary conditions) and the applied loads. The topology generation can thus be automated as part of the design optimization process.

Structural optimization problems often take into account some important design constraints such as weight constraints, assembly and packaging constraints etc. However, it is cumbersome to account for all design criteria (such as cost, economic factors, manufacturing, aesthetics etc.) simultaneously. The primary application of shape and topology optimization of structures may be to provide guidance to designers of structural components at the conceptual stage of design to select an optimal shape and topology based mostly on structural requirements. This design can then be modified at the detailing stage to account for manufacturing, aesthetic and other design criteria. By moving the design optimization iterations to an earlier (conceptual) stage in design, the designer can search a larger design domain and often avoid major modifications during the later stages when the cost of design modifications are higher. Modifications after the detailing stage can be very expensive and therefore design iterations shown by the loops on the right in figure 1.1 are to be avoided.

The objective of this research is to develop computationally efficient tools for structural optimization that enable the design of optimal shape and topology for mechanical components. The two primary goals of the research were to seek efficient shape representation schemes that enable both shape and topology variation and to develop optimization algorithms particularly suited for structural optimization. The major requirement of the shape representation scheme is that it allow shape variation not only by varying existing boundaries, but also by creating new boundaries to enable topology variation. Some important criteria for an optimization algorithm suited for structural optimization are that it should not require too many design analysis, should be able to handle large number of variables, should have a good convergence rate etc.

Shape optimization has been applied to the design of trusses, frames and shell structures as well as a variety of support structures that may be loaded in the plane or in three-dimensions. In this work, we model plane stress / plane strain loading situations so that designers can optimize the shape and topology of planar structures. The methods and algorithms developed in this thesis may be extended to shell structures and three-dimensional structures as well.

Progress in structural optimization research has paralleled progress in structural analysis methods. Since analysis is a prerequisite for structural optimization, development of sophisticated general purpose structural analysis tools have contributed to the development of more useful and powerful structural optimization techniques. Much of the early research in structural optimization was motivated by the need in aircraft and aerospace design applications to design light weight structures. This led to the development of methods for designing optimal trusses and frames and subsequently to the development of more general shape optimization techniques. Optimal design is becoming increasingly important in the automobile industry as standards for fuel economy get stricter. Most commercially available structural analysis tools now also include simple optimization capabilities. It is expected that very general shape and topology optimization techniques would become part of the design cycle of automobile panels and structural members in the future, just as the use of computational analysis techniques have become standard practice over the last two decades.

1.2. Design Optimization

Design optimization involves seeking the best design among a family of designs described using some parametrization. The parameters that are varied to obtain different members of the family (or design space) are referred to as the *design variables*. The designs in the design space are evaluated by some measure of performance. This criteria for determining the relative merit of the designs is referred to as the *design objective*. The design objective is described as a function of the design variables. This function is called the objective function in the optimization literature. In mechanical design applications, such functions are often non-trivial and the evaluation of these functions requires some kind of design analysis. Each design variable can vary within a range. The bounds on the variables that define this feasible range are referred to as the *side constraints*. These bounds are set to eliminate designs that may be infeasible from engineering considerations. In addition to the side constraints there may be several functional requirements represented as *design constraints* that must be satisfied by all valid designs. In a design optimization problem statement, these design constraints are expressed as functions of the design variables. These design constraint functions are also often nonlinear and may require numerical simulation to evaluate. The set of all the points in the parameter or design variable space that lead to valid designs is called the *feasible design domain*..

A design optimization problem such as structural optimization can be stated as follows:

Minimize (or Maximize): Design objective
subject to :
Design constraints
Side constraints on the variables.

The design objective and constraints are expressed as functions of the design variables. The design objective (or objective function) is often a nonlinear function in the mechanical and structural design context. We shall denote the objective function as $f(\mathbf{x})$, where $\mathbf{x}$ is the vector of design variables of dimension n, $\mathbf{x} \in R^n$. The design constraints may be equality or inequality functions of the design variables. We shall denote equality constraints as $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, $\mathbf{h}: R^n \rightarrow R^m$ to represent *m* equality constraints and the inequality constraints as $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$, $\mathbf{g}: R^n \rightarrow R^r$ to represent *r* inequality constraints. These functions may be linear or nonlinear. Those inequality constraints that are satisfied as an equality at the optimal solution are called *active constraints*. Those that are not satisfied as an equality are passive constraints and are essentially redundant constraint in the sense that removing them from the problem statement will not change the optimal solution. Ideally one would like to identify all the active constraints among the inequality constraints and eliminate all passive constraints. However, in most practical optimization problem it is not possible to predict which constraints are active without knowing the optimal solution, especially when there are a large number of inequality constraints. When the objective function and all the constraints are linear functions of the design variables, the optimization is referred to as a linear programming problem. If either the objective function or any of the constraints are nonlinear functions then the optimization is a nonlinear programming problem. Using the above notation a general optimization method may be stated as:

$$\begin{aligned} &\min \ f(\mathbf{x}) \\ &\text{subject to :} \\ &\mathbf{h}(\mathbf{x}) = \mathbf{0}, \ \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ &l_i \leq x_i \leq u_i \end{aligned} \tag{1.2.1}$$

Many algorithms exist to solve linear and nonlinear programming problems. Such optimization problems arise in a variety of fields so that linear and nonlinear programming techniques have a wide range of applicability. Accordingly, the literature is vast and innumerable algorithms have been proposed to solve these problems. The algorithm that is best suited to solve a particular optimization problem depends on the nature of the objective and constraint functions. A good review of nonlinear programming algorithms used in structural optimization has been compiled by Morris (Morris_82) and Haftka and Gurdal

(Haftka_92). In chapter 2, nonlinear programming algorithms have been classified in terms of their applicability to structural optimization problems. Brief descriptions of the algorithms are also given in this chapter.

1.3. Structural Synthesis

Structural optimization involves optimizing an objective function (often a structural property) while satisfying structural and other design constraints. Typical objectives for structural design are to improve the strength to weight ratio or to maximize stiffness. The functions involved are often nonlinear leading to a nonlinear programming problem. An optimization algorithm is used to solve the problem. The optimization algorithm iteratively modifies the shape and searches for feasible designs that better satisfy the stated objective.

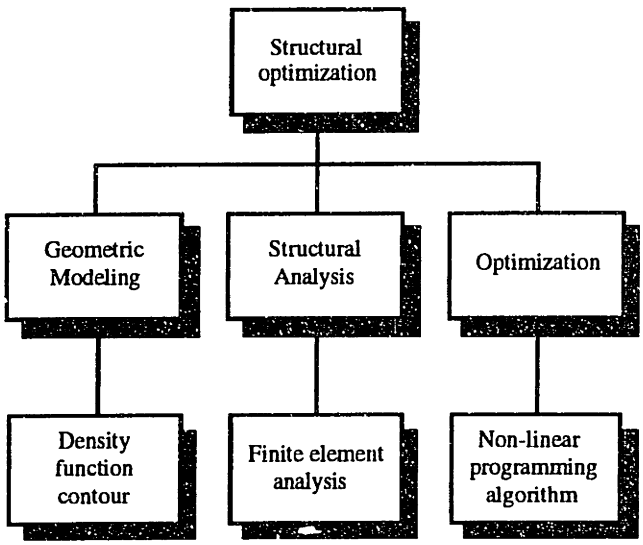


Figure 1.3. Structural optimization overview

Shape synthesis for structural components, when stated as an optimization problem, has three requirements. It needs a geometric modeling tool to represent the shape, a structural analysis tool and an optimization algorithm, as illustrated in figure 1.3. The geometry of the structure is the variable of the design optimization. The shape representation technique used depends on the kind of geometric variation desired. Structural optimization techniques can be classified on the basis of the type of design variables used to describe the geometry. The objective and the design constraints have to be expressed as functions of these design variables.

The optimization algorithm searches for the optimal value of these design variables. It evaluates the design at each iteration by evaluating the objective and constraint functions

and then determines how much to change the design variables (that is, modify the geometry). For simple structural design problems, it is often possible to derive analytical relations for the objectives or constraints as functions of design variables. However, most 2D and 3D structural optimization requires general purpose numerical analysis techniques such as the finite element method. Alternately, some researchers have used boundary element method (BEM) for structural analysis. However, these numerical methods are computationally intensive and hence function evaluation is expensive for structural optimization problems. Therefore, for this application, optimization algorithms that require very few function evaluations are desirable.

For a given geometry of the structure with applied loads and boundary conditions, the structural behavior can be expressed as a boundary value problem in elasticity. The shape of the structure is the domain of structural analysis and is denoted by Ω. The boundary of this domain is denoted by Γ. The part of the boundary that is fixed or supported is denoted by Γ_o and the part on which external forces or traction acts is denoted by Γ_t. This notation is illustrated in figure 1.4. The response of the structure under the action of these loads and boundary conditions can be obtained by solving the equilibrium equations and compatibility equations together with the linear elastic constitutive equations that describe the behavior of the material. The response of structures with arbitrarily complex geometry Ω under the application of design loads can be solved using numerical methods such as the finite element method.

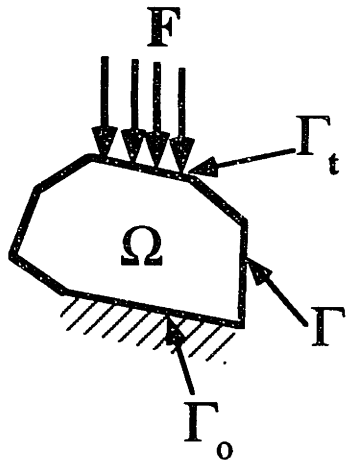


Figure 1.4. Boundary value problem in elasticity

The geometry of the structure Ω, is the variable for structural optimization. This geometry is parameterized so that it varies as a function of a set of design variables. The functions describing structural behavior such as the deflection of the structure due to loads,

the internal energy absorbed by the structure, etc. are functions of these design variables describing the geometry. When the design variables and therefore the geometry of the structure are modified, such functions would also get modified. For example, when the cross-sectional area of a bar is reduced, it would deflect more under a given tensile load. Sensitivity analysis techniques can be used to evaluate the sensitivity of the function f with respect to the design variables, such as the gradient vector ∇f, which is the vector of first partial derivatives of the function with respect to the design variables. The matrix of second partial derivatives of the function with respect to the design variables is called the hessian matrix, denoted as $\nabla^2 f$. Most optimization algorithms need to evaluate the sensitivity of the structural behavior with respect to these variables. When the number of design variables is very large, the hessian matrix becomes large, requiring a large amount of storage and expensive computation to assemble and use. Therefore, optimization algorithms that do not require the hessian matrix are preferred for structural optimization.

Based on the type of geometric variation allowed, structural optimization has been broadly classified into sizing, shape and topology optimization. A detailed survey of literature on sizing and shape optimization was documented by Haftka and Grandhi [Haftka_86].

Sizing optimization is applicable to the design of trusses, frames and membranes. The variables of the design are the cross-sectional areas of truss elements, thickness of membranes, etc. Here the layout of the truss and the shape of the membrane are not allowed to change.

Unlike sizing problems, shape optimization involves a change in the shape or the domain Ω of the structural analysis. Certain boundaries are specified as variable and defined using a parametric representation such as Bezier or B-Spline interpolation. The optimization algorithm modifies the shape by varying the shape of these boundaries.

Topology optimization involves geometric variation of a component, not only by boundary variation, but also by changing its layout by creating internal holes and therefore new boundaries. Topology variation allows global optimization of shape so that the final shape does not depend on the initial guess.

1.4. Problem Statement

The primary design goals of the structural design problem in this work are to maximize stiffness and minimize weight of a mechanical component while avoiding interference. To achieve this goal the objective and the constraints of the optimization problem are stated as :

Maximize Stiffness (or Minimize Strain Energy absorbed)
subject to :
Weight of structure $\leq$ Wo

The final shape of the component is also constrained to lie within a feasible region to avoid interference or spatial conflict with other parts with which it must be assembled. The shape is modified by the optimization algorithm by removing material from regions where material is under-utilized (i.e., not fully loaded) so that the final shape is a structure of maximum stiffness for the given weight. The stiffness of the structure can be maximized by minimizing the strain energy absorbed by the structure under the applied loads. The strain energy absorbed by the structure is a measure of the compliance of the structure since structures that are more compliant absorb more energy under a given load. Stiffer structures deform less under the application of load and hence absorb lesser energy. Therefore, a structure that has the minimum compliance deflects the least under a given load and hence are the stiffest. Details of the formulation of the objective and constraints are described in chapter 3.

1.5. Scope of Work

In this thesis, a methodology is described for the design of structurally optimal mechanical components. A new shape representation has been developed and used for structural optimization that allows both shape and topology variation. The optimization is carried out using a novel sequential approximation technique for nonlinear programming.

Chapter 2 gives a survey of the structural optimization literature. Section 2.2 briefly summarizes structural analysis using the finite element method as well as sensitivity analysis techniques to evaluate the gradient of the structural properties with respect to design variables. In section 2.3, the structural optimization problem is broadly classified into sizing, shape and topology optimization and the differences between them are summarized. Section 2.4 provides a classification of the nonlinear programming algorithms

and describes some algorithms that have been used for structural optimization and their computational implications.

The structural optimization problem statement is defined in Chapter 3. Section 3.2 describes the shape representation scheme we use and how it differs from some of the other popular techniques. The objective function and the constraints are described in section 3.3. The structural properties of the component are related to shape density function by assuming a relation between the material properties and the density function. This relation is described in section 3.4. In section 3.5, the implementation of the compliance minimization problem using finite element method is described. Section 3.6 gives a derivation of the gradient of the compliance with respect to the design variables.

Chapter 4 describes the nonlinear programming algorithm used to solve the structural optimization problem. Section 4.2 of this chapter is an overview of the philosophy of sequential approximate optimization algorithms along with a brief summary of some of the popular algorithms in this class that have been proposed for structural optimization. The algorithm used in this thesis will be referred to as the moving barrier sequential linear programming (or MBSLP) and in section 4.3 we describe its application to optimization problems that have only linear constraints. In section 4.4, a technique for determining an initial feasible point for the optimization problem is described. The algorithm has been extended to solve optimization problems with nonlinear constraints in section 4.5.

Details of the implementation of the shape and topology optimization software is described in chapter 5. Section 5.2 describes the shape and topology optimization algorithm and its implementation. The implementation of the sequential approximate optimization algorithm used in this thesis is described in section 5.3.

Chapter 6 contains examples illustrating the application of the algorithms developed in this thesis. The optimization algorithm has been applied to many nonlinear programming problems in section 6.2 to test its performance. Section 6.3 gives some examples and applications of the shape and topology optimization algorithm to design optimal structures.

Chapter 7 is the concluding chapter that gives a summary of the thesis as well as recommendations for future work.

2 Previous Work in Structural Optimization

2.1. Overview

In structural optimization the variable of the design is the geometry of the structure. The geometry or shape of the structure is parameterized to enable the required geometric variation. These parameters are the design variables of the problem. Many different design objectives and constraints have been used in structural optimization problems. The most common examples are to minimize the weight of the structure subject to constraints on the stress or maximize stiffness subject to constraints on weight.

Early work in structural synthesis involved the design of the cross-sectional areas of truss and frame members [Kirsch_81, Morris_82, Topping_83]. The design variables are the cross-sectional areas of the members of the truss or frame. Such problems are relatively easy since the number of variables are small and the analysis of these structures very inexpensive. A similar application is the design of optimal plates where the thickness is variable. In this case finite element analysis is used to analyze the structural properties. The design variables are the thickness of each plate element.

Significant progress in structural optimization occurred with the introduction of boundary variation [Briabant_84, Haftka_86]. In this case, the shape is parameterized such that its boundaries vary with these parameters. Examples of this situation are when the radius of a circular hole or the width of a rectangular hole is treated as the design variable or when Bezier or B-spline curves are used to parametrize the boundary. In this case, the number of design variables may be small, however, when finite element analysis is used to analyze the structure, the change in shape implies the need for recreating the mesh at each iteration.

Shape optimization by boundary variation does not always lead to truly optimal designs. The parametrization allows the variation of certain boundaries, but does not allow

the creation of new boundaries. As a result the topology of the geometry does not change during the optimization process and if the designer does not correctly guess the optimal topology for the initial geometry, sub-optimal shapes are obtained. The techniques that have been developed for topology optimization treat the material as a composite or a porous material [Kohn_86, Bendsoe_88]. The optimization problem then involves solving for the optimal pore distribution (or equivalently the optimal material distribution). These techniques use homogenization techniques to determine the material properties of the porous material. Topology optimization problems usually require a large number of design variables to characterize the distribution of pores.

Nonlinear programming algorithms are used to compute the optimal solution of the structural optimization problem. In the design of trusses and frames, where the analysis involved is inexpensive, most nonlinear programming algorithms perform well. However for the design of general 2D and 3D structures, costly numerical methods are required for structural analysis. Therefore, algorithms that tend to use fewer function and gradient evaluations have been favored in structural optimization. Sequential approximation algorithms typically use fewer function and gradient evaluations and as a result have been popular in structural optimization [Schmit_74, Fluery_79, Haftka_92] .

Structural synthesis by optimization requires very general and automated structural analysis techniques. Section 2.2 provides a brief introduction to the finite element method to analyze structures. In this section, we also describe sensitivity analysis techniques that are used to obtain the gradient of structural properties with respect to design variables. In section 2.3, structural optimization is classified on the basis of the design variables used to parameterize the design. It is also possible to classify the literature on the basis of the design objectives and constraints. The commonly used objectives and constraints are also described in this section. Optimization problems in structural synthesis applications are often nonlinear. An introduction to nonlinear programming algorithms used to solve such optimization problems is given in section 2.4.

2.2. Structural Analysis

In this section a brief introduction to the finite element method for structural analysis is presented, followed by a review of techniques used to find the design sensitivity of structures that can only be analyzed using numerical methods such as finite element method.

2.2.1. The Finite Element Method

With the exception of a few simple truss or frame like structures, most structural optimization techniques use general purpose numerical methods such as finite element method or boundary element method for structural analysis. In this section, a brief description of the finite element analysis of plane strain - plane stress linear boundary value problem is presented to clarify the notations used in this thesis. Details about the finite element method can be found in text books such as [Bathe_82, Reddy_84, Kikuchi_86, Zienkiwicz_89].

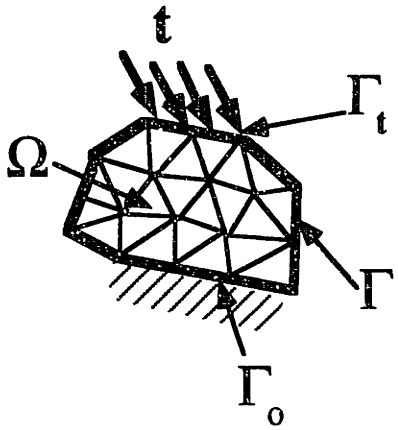


Figure 2.1. Planar structure with applied loads and boundary conditions

Consider a structure of arbitrary shape shown in figure 2.1. The domain of the structure Ω is divided into a finite element mesh shown as a triangular mesh in the figure. The structure is subjected to a body force $\mathbf{f}$ and a traction $\mathbf{t}$ along part of its boundary. The principle of virtual work that governs the behavior of the structure may be expressed as,

$$\int_{\Omega} \sigma \delta\varepsilon \, d\Omega = \int_{\Gamma_t} \mathbf{t}.\delta\mathbf{u} \, d\Omega + \int_{\Omega} \mathbf{f}.\delta\mathbf{u} \, d\Omega \qquad (2.2.1)$$

σ is the stress field created in the structure due to the applied forces and boundary conditions. $\delta\mathbf{u}$ and $\delta\varepsilon$ are the virtual displacement and the corresponding virtual strain respectively, so that the left hand side of the above equation represents the virtual internal energy absorbed by the structure and the right hand side represents the virtual work done by the external body forces and the traction. $\delta\mathbf{u}$, the virtual displacement field, must be kinematically admissible subset of Sobolev space, so that, $\delta\mathbf{u} \in H^1(\Omega)$, $\delta\mathbf{u} = \mathbf{0}$ on Γ_o.

The relation between the stress and the strain is given by the constitutive equations. For a linear elastic material, these equation are given by Hook's Law and can be written as,

$$\{\sigma\} = [\mathbf{D}]\{\varepsilon\} \tag{2.2.2}$$

where, $\{\sigma\}$ and $\{\varepsilon\}$ are the stress and strain fields expressed as vectors. For planar structural analysis,

$$\begin{aligned} \{\sigma\} &= \{\sigma_{xx}, \sigma_{yy}, \sigma_{xy}\}^t \\ \{\varepsilon\} &= \{\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}\}^t \end{aligned} \tag{2.2.3}$$

$$[\mathbf{D}] = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{2.2.4a}$$

where, $[\mathbf{D}]$ is the elasticity matrix for plane stress, E is the Young's modulus of elasticity and ν is the Poisson's ratio. The elasticity matrix for plane strain is given by,

$$[\mathbf{D}] = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \tag{2.2.4b}$$

The strain vector can be calculated from the displacement vector field $\mathbf{u} = \{u_x, u_y\}^t$ as,

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x},\ \varepsilon_{yy} = \frac{\partial u_y}{\partial y},\ \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) \tag{2.2.5}$$

The integral equation representing the principle of virtual work is integrated piece-wise and the unknown displacement fields $\mathbf{u}$ and $\delta\mathbf{u}$ are represented by a piece-wise continuous interpolations. This converts the integral equation into a set of linear simultaneous equation in terms of the coefficients of the interpolation functions used to represent $\mathbf{u}$. These equations are solved using standard linear equation solvers such as LU decomposition [Press_88]. The integral equations are reduced to a set of linear algebraic equations of the form,

$$\{\delta\mathbf{u}_h\}^t[\mathbf{K}]\{\mathbf{u}_h\} = \{\delta\mathbf{u}_h\}^t\{\mathbf{F}\} \ \Rightarrow\ [\mathbf{K}]\{\mathbf{u}_h\} = \{\mathbf{F}\} \tag{2.2.6}$$

where, $[\mathbf{K}]$ is the global stiffness matrix, $\{\mathbf{F}\}$ is the resultant external force vector and $\{\mathbf{u}_h\}$ is the discrete representation of the displacement field and it consists of the coefficients of the piecewise-interpolation used to represent the displacement field $\mathbf{u}$.

In this thesis triangular finite elements were used and the displacement fields **u** and **δu** were linearly interpolated with each triangle as follows,

$$\begin{aligned} u &= u_1L_1 + u_2L_2 + u_3L_3 \\ v &= v_1L_1 + v_2L_2 + v_3L_3 \end{aligned} \tag{2.2.7}$$

where, L_1, L_2 and L_3 are shape functions for linear interpolation, also referred to as area coordinates or Barycentric coordinates [Zienkiewicz_89]. These coordinates are defined by the equations (2.2.7) and the assumption that $L_1+L_2+L_3=1$. The transformation from the (x,y) coordinates to (L_1,L_2,L_3) coordinates can be obtained as

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \frac{1}{2\Delta} \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{2.2.8}$$

where,

$$a_i = x_j y_k - x_k y_j, \tag{2.2.9}$$

$$b_i = y_j - y_k \tag{2.2.10}$$

$$c_i = x_k - x_j \tag{2.2.11}$$

and i, j and k are cyclically allotted the values 1, 2 and 3. Δ is the area of the triangular element and (x_i,y_i) are the nodal coordinates of the triangular element.

2.2.2. Sensitivity Analysis

Most nonlinear programming algorithms used to compute the optimal values of design variables require sensitivity information such as the gradient of the functions with respect to the design variables. Sensitivity analysis can be very expensive and therefore, considerable research effort has been directed at developing efficient techniques for sensitivity analysis of functions that represent structural behavior. A comprehensive description of design sensitivity analysis techniques for structural systems has been compiled by Haug *et al* [Haug_86].

The simplest method for finding the sensitivity is to use finite difference approximation for all the partial derivatives. The idea is to evaluate the function for two values of the design variable very close to each other and then to approximate the partial derivative as the ratio of the change in the function to the change in the variable. For example, the forward-difference approximation is given as

$$\frac{\partial f}{\partial x_i} \approx \frac{\Delta f}{\Delta x_i} = \frac{f(x_i + \Delta x_i) - f(x_i)}{\Delta x_i} \tag{2.2.12}$$

where, f is the function, x_i are the design variables and Δf is the change in the function due to the change Δx_i in the design variable x_i.

Even though the finite difference approximation is easy to implement, it is very computationally expensive for sensitivity evaluation of structural behavior. In order to evaluate functions that represent structural behavior, it is often necessary to perform a structural analysis. This can be very expensive when the analysis is to be performed by numerical methods. To evaluate the gradient of a function using the finite difference approximation described above, each gradient evaluation requires two function evaluations.

It is often possible to directly differentiate the governing equations of the structural analysis to obtain sensitivity information. Such an approach is particularly straight forward when the domain of the structural analysis Ω does not change with the design parameters and only the stiffness varies. This is the case for sizing optimization and also for topology optimization using homogenization method as well as the methods proposed in this thesis. Consider a function $g(\mathbf{u},\mathbf{x})$ that represents a structural behavior such as its compliance or a critical stress. This function may depend directly on the vector of design variables $\mathbf{x}$ and also indirectly due to its dependence on the displacement field $\mathbf{u}$. The displacement field expressed in the discretized vector form $\{\mathbf{u}_h\}$ in equation (2.2.6) is, of course, a function of the design variables. Differentiating this function with respect to the design variables, we get,

$$\frac{dg}{dx_i} = \frac{\partial g}{\partial x_i} + \frac{\partial g}{\partial \mathbf{u}_h}\frac{\partial \mathbf{u}_h}{\partial x_i} \tag{2.2.13}$$

where, $\frac{\partial g}{\partial \mathbf{u}_h}$ is a vector of elements $\frac{\partial g}{\partial u_{hj}}$ and $\frac{\partial \mathbf{u}_h}{\partial x_i}$ is a vector of elements $\frac{\partial u_{hj}}{\partial x_i}$, $j=1,..n_a$, where n_a is the number of degrees of freedom in the finite element model.

Differentiating the governing equations expressed in a discrete form in equation (2.2.6), we get,

$$\mathbf{K}\frac{\partial \mathbf{u}_h}{\partial x_i} = \frac{\partial \mathbf{F}}{\partial x_i} - \frac{\partial \mathbf{K}}{\partial x_i}\mathbf{u}_h \tag{2.2.14}$$

Equation (2.2.14) can be solved for $\frac{\partial \mathbf{u}_h}{\partial x_i}$ and substituted into (2.2.13) to obtain the gradient of the constraint g(**u**,**x**). This is the 'direct approach'. Alternately, the so called adjoint method can be used where a set of adjoint variables λ_a is defined by,

$$\mathbf{K}\lambda_a = \frac{\partial g}{\partial \mathbf{u}_h} \tag{2.2.15}$$

so that, the gradient of the function g, can be written as,

$$\frac{dg}{dx_i} = \frac{\partial g}{\partial x_i} + \lambda_a \left(\frac{\partial \mathbf{F}}{\partial x_i} - \frac{\partial \mathbf{K}}{\partial x_i} \mathbf{u}_h \right) \tag{2.2.16}$$

Equation (2.2.15) may be solved to obtain λ_a and its value can be substituted into equation (2.2.16) to get the desired gradient. When the number of variables is larger than the number of functions whose gradients have to be found, the adjoint method is more efficient. For the direct method equation (2.2.14) must be solved once for each variable, whereas for the adjoint method equation (2.2.15) needs to be solved only once for each function whose gradient is to be evaluated.

The methods described above assumes that it is easy to compute the derivative of the stiffness matrix with respect to the design variables ($\partial \mathbf{K}/\partial x_i$.) However, this may not be the case, especially if one does not have access to the source code of the finite element analysis software. It is possible to directly differentiate the principle of virtual work (2.2.1) instead of the discretized version (2.2.6). This leads to the variational sensitivity analysis techniques. The details of this method can be found in [Haug_86] and [Haftka_92].

In the sensitivity analysis techniques described so far we have assume that the shape of the domain of analysis does not change. However, during shape optimization, the shape of the structure and therefore the domain of the analysis changes. The scalar and vector fields defined over the domain change with the design variable not only because of the change in structural behavior with the shape but also because the deformation of the domain transforms the coordinates used to define the field. This leads to the concept of material derivative, analogous to the material derivatives defined in fluid mechanics. Let the initial domain be Ω_0 and the modified domain be Ω. Let T be the transformation mapping from the coordinates $\mathbf{X}_0$ from the initial domain to the coordinates **X** in the modified domain, so that $\mathbf{X} = T(\mathbf{X}_0, \mathbf{x})$ and $\Omega \equiv T(\Omega_0, \mathbf{x})$. The *design velocity* is defined as the rate at which the domain is transformed with change in design variables x_i,

$$\mathbf{V}_i = \frac{\partial T}{\partial x_i}, \quad i=1,..n \tag{2.2.17}$$

The material derivative of a scalar field u(**X**,**x**), may then be defined as,

$$\frac{du}{dx_i} = \frac{\partial u}{\partial x_i} + \mathbf{V}_i \nabla u, \text{ where, } \nabla u = \left\{ \frac{\partial u}{\partial X_i} \right\} \tag{2.2.18}$$

Similarly, material derivatives can be defined for functionals defined in terms of the scalar of vector fields defined over the domain. Typically, the transformation of the domain with respect to the design variables is defined using the concept of design elements described in section 2.3.2 on shape optimization. Even though sensitivity analysis for varying structural domains is well established, the added computational expense and complexity associated with it is one of the draw backs of shape optimization by boundary variation.

2.3. Structural Optimization

Structural optimization may be classified on the basis of the design variables chosen to describe the geometry. They may also be classified based on the particular combination of objective function and constraints used to describe the optimization problem. Based on the type of design variables used, structural optimization may be classified as sizing, shape and topology optimization. These classifications and the type of geometric parametrization used are described below.

2.3.1. Sizing Optimization

The earliest attempts at structural optimization used simple parametrization of the geometry and optimized easy-to-analyze structures such as trusses, frames and plates [Haftka_86, Haftka_92, Kirsch_81, Morris_82] . In sizing optimization, the variables of the design are the cross-sections of the truss members or thickness of the plates etc. The geometry change induced by varying these design variables is not drastic so that there is no need to create a new analysis model each time the design variables are changed. Such design variables are called sizing variables.

Trusses and Frames

Structural optimization of trusses and frames involves the optimal design of the cross-sections of the elements. The design variables are the cross-sectional areas of the truss or

frame elements. For simple trusses with only a few members, analytical expressions may be derived to express the structural properties as a function of the design variables. Larger trusses and frames can be analyzed using the finite element method. Cross-sectional dimensions of the truss elements are sizing variables since the finite element model or the analytical expression describing the behavior of the truss is valid for all feasible values of these variables. Figure 2.2 illustrates a typical truss design problem with the applied loads and boundary conditions. The nodal coordinates and the lengths of the elements are fixed for the cross-section design problem.

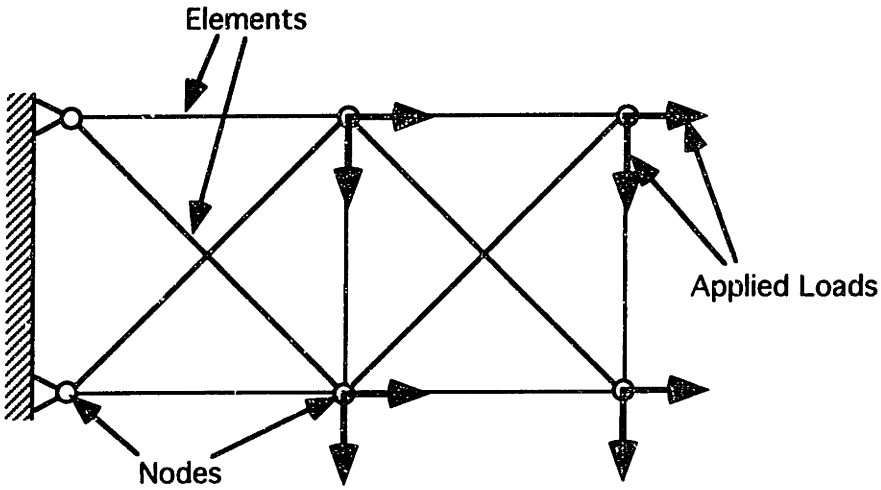


Figure 2.2. Truss with applied loads and boundary conditions

Trusses and frames can be optimized also by varying their configuration. The optimal configurations of a truss can be computed by solving for the optimal nodal coordinates. In this case, the design variables are the nodal coordinates of the truss. The topology or connectivity of the truss is fixed and therefore, for configuration design there is again no need to modify the analysis model when the variables are changed.

Yet another choice of variables for the optimal design of trusses are material selection parameters that distinguish which material to use for each element of a truss or frame. Material selection is a combinatorial optimization problem of selecting the optimal material, from a given set, for each element of the truss or frame. In truss and frame optimization, it is common to simultaneously consider a combination of these three types of sizing variables. For example, the cross-sectional areas of the elements and coordinates of the nodes may be varied simultaneously. Figure 2.3 shows an example of a 39 bar truss whose configuration as well as the cross-sectional areas of its elements are optimized. This example has been adapted from [Morris_82]. Figure 2.3(a) illustrates the initial configuration of the truss while figure 2.3(b) show the optimal configuration and cross-

sections for supporting the load acting on nodes 13, 14, and 15 as shown. Details of this example as well as the algorithm used for the optimization can be found in [Morris_82].

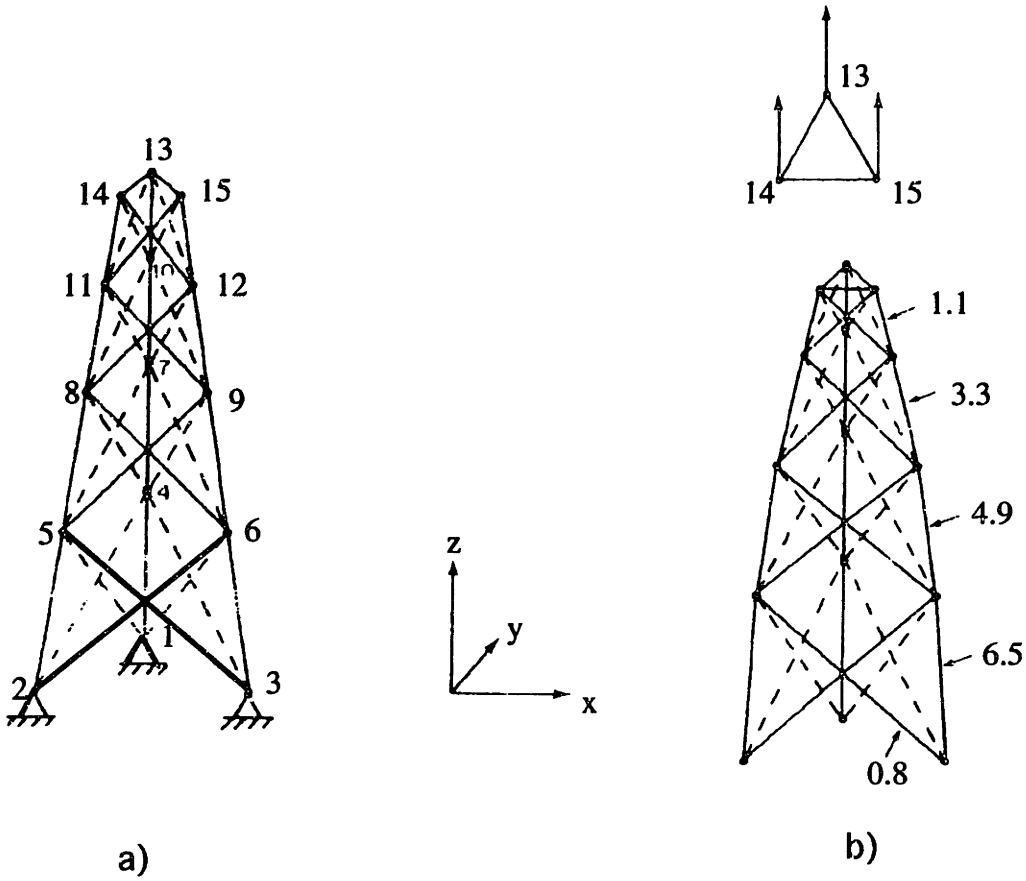


Figure 2.3. 39 bar truss (Adapted from Moris_82)

Plates

In early attempts at designing optimal plate-like structures, the plate thickness was chosen as the design variables. This choice of variable makes the problem a sizing optimization, since the overall shape or topology of the plate remains fixed during the optimization and only its thickness is allowed to vary. The analysis model therefore, does not change during the optimization. The thickness is assumed constant within each element but can vary from one element to the other. The thickness of each element is treated as a design variable. The results obtained by such optimization are not satisfactory, since the thickness varies very discontinuously from element to element and the mesh that is adequate for the finite element analysis is often not adequate for representing the optimal thickness distribution.

2.3.2. Shape Optimization

Shape optimization as opposed to sizing optimization involves varying the boundaries of the structure thus modifying the domain of structural analysis. As a result, shape optimization requires the finite element model to change during the optimization process. The design variables used for shape optimization cause boundary variation and are referred to as the *shape design variables.* Boundary variation is usually expensive relative to sizing optimization due to the changing finite element model and because more sophisticated sensitivity analysis tools are required. However, shape optimization is more general and can often significantly improve the performance of 2D and 3D structures.

Parametric Variables

Shape design variables could be parameters defining certain features of the shape or important dimensions. For example, the radius of a circular hole or the side of a square hole could be a design variable. The length or breadth of structural components can also be treated as shape design variables. Clearly changing these parameters can significantly change the geometry and in some cases require the regeneration of the mesh. Many examples of shape variation using parametric variables have been illustrated by Botkin et. al [Botkin_86]. An examples of parametric variables is shown in figure 2.4. The radii x_1 and x_2 and the dimensions of the slot are design variables. When these variables are changed, the shape changes but the topology does not. The finite element mesh needs to be adjusted or recreated when the design variables are modified. A number of geometric constraints are imposed to maintain the validity of the shape. For example, the edges joining the semi-circles of radii x3 and x4 should remain tangent to these semi-circles when the radii are modified.

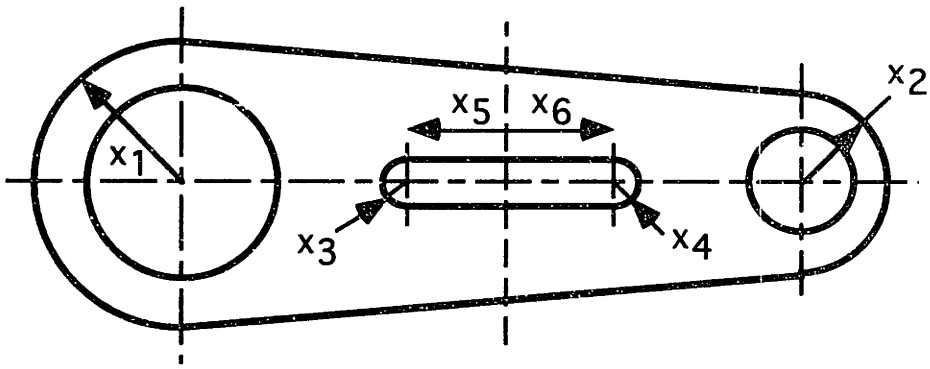
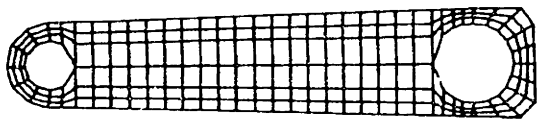


Figure 2.4. Parametric design variables

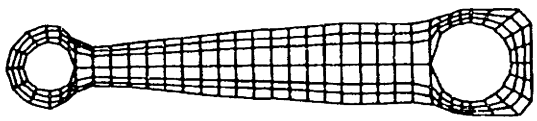
Boundary Variation

Shape optimization can also be achieved by treating parts of the boundary of a solid as variables by choosing shape design variables that parameterize parts of the boundary. For

example, the nodal coordinates of the nodes on the boundary of the shape may be treated as the design variables. However, one important criterion for shape optimization is that the finite element model should not deteriorate during the optimization. This may require regeneration of the finite element mesh at each iteration. Therefore, a one-to-one correspondence between the finite element mesh and the design variables is not desirable for shape optimization. Figure 2.5 shows an example of shape optimization. Figure 2.5(a) shows the initial shape of a torque arm selected by the designer. Parts of the outer boundary are treated as variable and the shape is optimized to obtain the final design in figure 2.5(b). Notice that the finite element mesh has modified with the shape. Details of this example can be found in [Yang_86].



(a)



(b)

Figure 2.5 Example of shape optimization (Adapted from Yang_86)

In shape optimization by boundary variation, the concept of a design element is often used [Imam_82, Briabant_84]. Design elements are regions into which a structure is divided and the boundaries of each such region is controlled by a set of design variables. Each design element may consist of many finite elements. Many methods of parameterizing the boundaries have been used and they are described below. Early approaches used the two-dimensional isoparametric interpolation functions used in the finite element formulation to represent the boundaries of the design elements. Therefore, the positions of nodes of the isoparametric finite elements were the shape design variables. However, as described earlier, this one-to-one correspondence between the finite element mesh and the design variables was found to be undesirable. Figure 2.6. illustrates the concept of design

elements. Each design element may be divided into many finite elements for analysis purpose, as shown in figure 2.6 for design element 2. The shape of each design element is controlled by a set of shape design variables. Parametric variables such those described in the previous section have also been used to vary the shape of design elements.

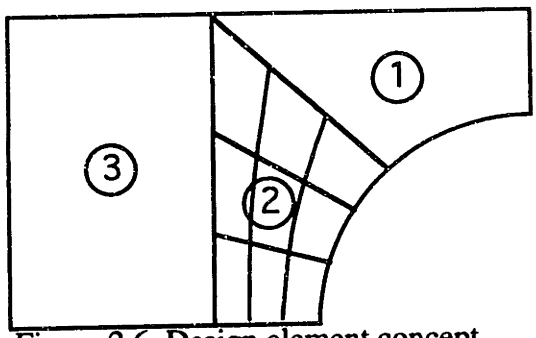


Figure 2.6. Design element concept

Many researchers [Haftka_86] have proposed the use of polynomial functions to represent the boundaries of the design elements. The polynomial coefficients may then be the shape design variables. More generally the boundary could also be treated as a linear combination of shape functions. The weighting coefficients could then serve as the design variables. However, the use of very high order polynomials can lead to oscillatory boundaries.

Briabant and Fleury [Briabant_84] have used cubic splines to represent the boundaries. The structure is divided into design elements and blending functions commonly used in computer graphics, such as Bezier or B-spline curves and surfaces, are used to represent the design element boundaries. The control points of these blending functions then serve as the variables of the structural optimization problem. A design element parameterized using b-spline blending functions is illustrated in figure 2.7.

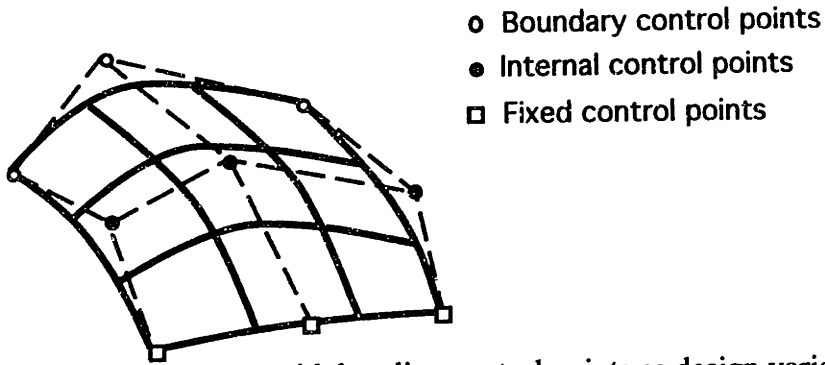


Figure 2.7. Design element with b-spline control points as design variables

In figure 2.7, the coordinates of the boundary control nodes are the design variables. The boundary of the design element is modified when the boundary control points are

moved . The boundary marked by fixed control nodes remains unaffected by changes to the boundary control points. The internal control nodes define the isoparametric curves on the design element that are used to define the mesh. When the boundary control nodes are moved, the internal control points are moved homothetically along meridians such that the mesh density remains nearly uniform within the design element.

The use of Bezier or b-spline blending functions provides greater flexibility in representing the boundaries and can represent relatively complex shapes using fewer design variables. Splines are composed of low order piecewise polynomials and therefore do not have oscillatory behavior. They also maintain boundary regularity and smoothness. Some properties that make Bezier and b-splines curves desirable are: the variation diminishing property, axis independence and the ability to represent multi-valued functions and closed curves. The variation diminishing property ensures that the curve will lie within the convex hull of the control points defining the curve. Axis independence ensures that the curve representation is independent of the coordinate axes used to locate the control points. The degree of the Bezier curve depends on the number of control points. It is possible to piece together many low order Bezier curves and enforce the desired degree of continuity at the joints. Bezier curves however, do not allow local control of the curve, so that the location of each control point affects the entire curve. B-splines on the other hand allow local control of the curve shape and its degree is independent of the number of control points. Consequently, b-spline representation is a preferred means of boundary representation for shape optimization. Analytical expressions can be derived for the sensitivity of the design objective and constraint functions with respect to the control point positions.

Technical difficulties

As the shape is modified during the shape optimization process, very often the finite element mesh may need to be refined or fully regenerated. Mesh refinement is required when the shape changes significantly so that the finite element mesh is highly distorted in certain regions causing loss in accuracy. If the shape changes drastically a new mesh may need to be generated. Since the shape is automatically modified by the optimization algorithm, mesh generation should also be automated. This is one of the major difficulties associated with shape optimization. However, at least in 2D, some good mesh generation and mesh refinement algorithms are now available.

Most optimization algorithms need sensitivity analysis to determine the rate of change of the objective and constraints with respect to the design variables. Sensitivity analysis for shape optimization is complicated by the fact that the domain of the analysis is changing, so

that the rate of change of the displacement field must account for not only the changes in design variables but the transformation of the local curvilinear coordinates of the design element due to the deformation of the domain. Sensitivity analysis for shape optimization is briefly described in section 2.2.2.

In addition to the above mentioned issues, there are some fundamental difficulties with the shape optimization. First, the final design depends on the initial guess, due to the inability of this approach to make modifications to the topology. As a result, it converges to different optimal shapes for different starting topologies. Furthermore, such optimization schemes are very sensitive to the accuracy of the structural analysis solution at the boundaries.

2.3.3. Topology Optimization

Due to the above mentioned limitations of shape optimization, it was realized that to obtain globally optimal shapes the topology must be modified, allowing the creation of new boundaries.

Early attempts towards topology optimization involved structural analysis using finite element method followed by removal of elements that were under-stressed. This approach was unsuccessful because the final shapes were found to depend on the initial mesh density used for the finite element analysis. Strang [Strang_86a] attributed this behavior to the non-convex nature of the problem statement. Kohn and Strang [Kohn_86], noting that the original problem was not well posed, suggested a *relaxed variational problem* that allows composites (or porous material) instead of the "0-1 dichotomy between holes and material". The properties of such composites can be derived using the homogenization technique. Homogenization is a mathematical tool that predicts the variation of material properties of a composite material due to variation in the amount of its constituents, in the limit when the size of individual particles of the constituents tends to zero.

Topology Design of Trusses

The earliest attempts to design topologically optimal structures was in the design of truss-like (or skeletal) structures. Research in this area has been extensive and a review of literature on the optimization of skeletal structures has been compiled by Topping [Topping_83]. The most common technique is the so called 'ground structure' approach. In this approach, the design space is covered with a grid of nodes. These nodes include locations where the loads are applied and boundary conditions imposed. The 'ground

structure' is constructed by connecting every node to every other node. Linear programming methods can be used to optimize this ground structure with the objective of minimizing the weight subject to constraints on the plastic collapse load. During the optimization all the unnecessary members of the ground structure are automatically removed when their cross-sectional areas are reduced to zero by the optimization algorithm yielding the optimal topology of the structure. It was found that in this approach the optimal structure is not unique even when the optimal weight is unique. When stress or displacement constraints and multiple-loads are to be considered, nonlinear programming techniques need to be used. Automatic removal of unnecessary members is no longer easy, since the stress in these members become large as their cross-sectional areas tend to zero. Another problem is that the stiffness matrix may become singular due to the removal of some members. Many methods for overcoming these difficulties have been proposed [Topping_83, Haftka_92].

The optimal layout or topology obtained by the 'ground structure' approach depends on the grid of nodes used to construct the ground structure. The grid that is used significantly influences both the layout and the optimum weight of the structure. The 'geometric' approach to layout design of skeletal structures treats both the nodal coordinates of the grid and the cross-sectional areas of the members as variable.

Homogenization Method

Homogenization is a mathematical tool for modeling the behavior of periodic structures, such as composites and porous materials, whose properties vary periodically at a microscopic level [Bensoussan_78]. The behavior of the structure at the macroscopic level is predicted in the limit when the ε_r (the ratio of period of the structure to a typical macroscopic length scale) tends to zero.

Mathematically, a boundary value problem for a structure may be stated in the form,

$$\mathbf{K}^{\varepsilon}\mathbf{u}_h = \mathbf{F},\ \mathbf{u} \text{ is kinematically admissible displacement field} \qquad (2.3.1)$$

$\mathbf{K}^{\varepsilon}$, the coefficients of the differential operator (stiffness matrix if FEM is used), depends on the periodicity of the structure. Stated differently, the material property of the composite structure would depend on the constituents and on the ratio of the constituents (ε_r). Homogenization predicts the behavior of the composite and yields an homogenized differential operator $\mathbf{K}_H$ such that $\mathbf{K}^{\varepsilon} \rightarrow \mathbf{K}_H$ as $\varepsilon_r \rightarrow 0$.

In the context of shape and topology optimization, Kohn and Strang [Kohn_86] proposed the use of homogenization as a means for allowing a uniform variation of properties from 0 to 1. Previous attempts at topology optimization had allowed only solid (1) and hole (0). The holes were created by removing under stressed elements. Such methods faced serious convergence difficulties. Mathematically, this suggested the need for relaxing the original variational problem statement so that it would be well-posed (allowing convergence). Kohn and Strang show that, relaxing the variational problem is identical to allowing composite materials. The properties of such a composite material can be predicted using homogenization.

Bendsoe and Kikuchi [Bendsoe_88] have carried the above idea further. They assume that the material is porous and solve for the optimal distribution of porosity. A design domain is defined as the space within which the structure has to fit. This domain is divided into a rectangular mesh. The loads to be carried by the structure and the support conditions are prescribed by the designer as shown in figure 2.8.

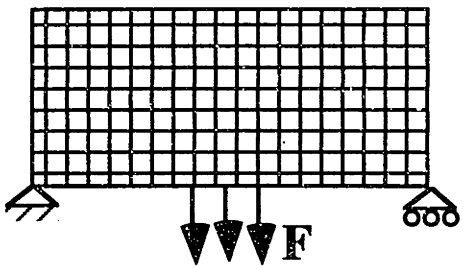


Figure 2.8. Design domain and applied loads

The structural behavior under the applied load and boundary conditions is analyzed using the finite element method. The design domain is taken as the initial shape of the structure. The material is modeled as porous by assuming a microstructure. A unit cell of the microstructure is shown in figure 2.9(a). The material is assumed to be made up of infinite such units cells that are assumed to be infinitesimal in the limit. Suzuki and Kikuchi [Suzuki_91] have assumed a rectangular void of dimensions 'a' and 'b' within the unit cell as shown in figure 2.9(a). The dimensions of the void within the unit cell determines the overall porosity or void fraction of the material. Each finite element is assumed to have a fixed porosity or void fraction so that it is associated with its own void dimensions a_i and b_i, where 'i' is the element number. These void dimensions along with the orientation of the unit cells in each finite element θ_i are taken as the design variables. Initially, the material is assumed to be uniformly porous everywhere in the structure so that all the elements are assumed to have the same porosity or void dimensions a_i and b_i and orientation θ_i.

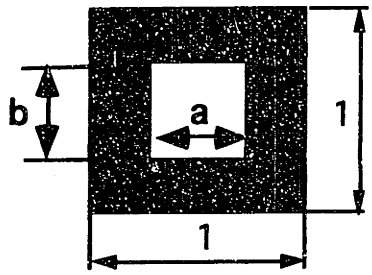


Figure 2.9.(a) A unit cell

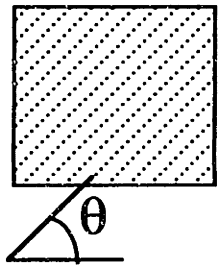


Figure 2.9.(b) Unit cell orientation in a element

An optimality criteria algorithm was used [Bendsoe_88, Suzuki_91] to solve for the optimal distribution of porosity, that is, the optimal value of void dimensions and orientations for each element. The properties of the material obviously depend on the assumed microstructure and therefore on the void size in each unit cell. The material properties are a continuous function of the void dimensions. For a given porosity or void size, the material properties can be determined using the homogenization method.

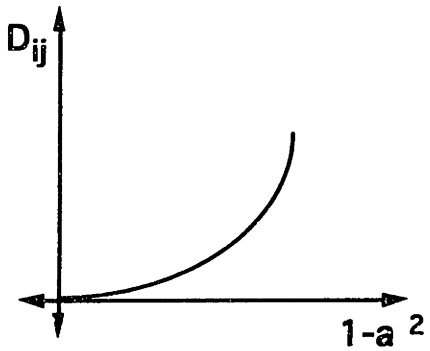


Figure 2.10. Material property as a function of density

Figure 2.10 illustrates the typical relation between a material property coefficient and pore/void size, where the void in the unit cell is assumed to be a square (a=b), so that, $1-a^2$ represents the density of the unit cell (and the material). When there is no void in the unit cell, that is a=0, then the material is fully dense and the material properties are the same as that determined experimentally for the non-porous material. As the pore size becomes larger, the density of the material decreases and the material property coefficients decrease. The homogenization method can be used to predict the material property coefficients D_{ij} for a given porosity (void size). A finite element analysis over the unit cell is required to evaluate the material property coefficient for each value of void size. Therefore, to obtain a relation such as that shown in figure 2.10, the value of D_{ij} is evaluated for a discrete number of values of the void dimension 'a' and then the functional relationship is obtained by interpolating these values using Legendre polynomials. For square and rectangular

voids highly nonlinear relationships are obtained. The evaluation of the relation in figure 2.10 therefore requires several finite element analysis over the unit cell. The relation obtained would be different if a different microstructure is assumed. When two variables are used to characterize the microstructure such as in figure 2.7(a) (ie, a≠b), two-dimensional interpolation is required to represent the material property versus void size (a,b) relation. To avoid extra computation while solving for the optimal void size distribution, the relation such as in figure 2.10 needs to be evaluated *apriori* and stored for each microstructure assumption.

The optimization algorithm automatically increases the pore size (or decreases density) for elements where material is under-utilized (under-stressed) and decreases the pore size where the material is highly utilized (over-stressed) and therefore needs to be strengthened. When the optimization algorithm converges, some elements would have very large void size so that they have very low density and others would have zero void size (fully dense material). The elements that have density below a certain threshold value are removed to obtain the geometry with the optimal topology. Typically, this threshold is set to zero or nearly zero density (or void size nearly equal to unit cell size).

As noted earlier, the material property density relation varies with the microstructure assumed for the composite or the porous material. In the method adapted by Bendsoe and Kikuchi [Bendsoe_88] square or rectangular void was assumed within a unit cell. More recently Allaire and Kohn [Allaire_92] have used rank 2 microstructure.

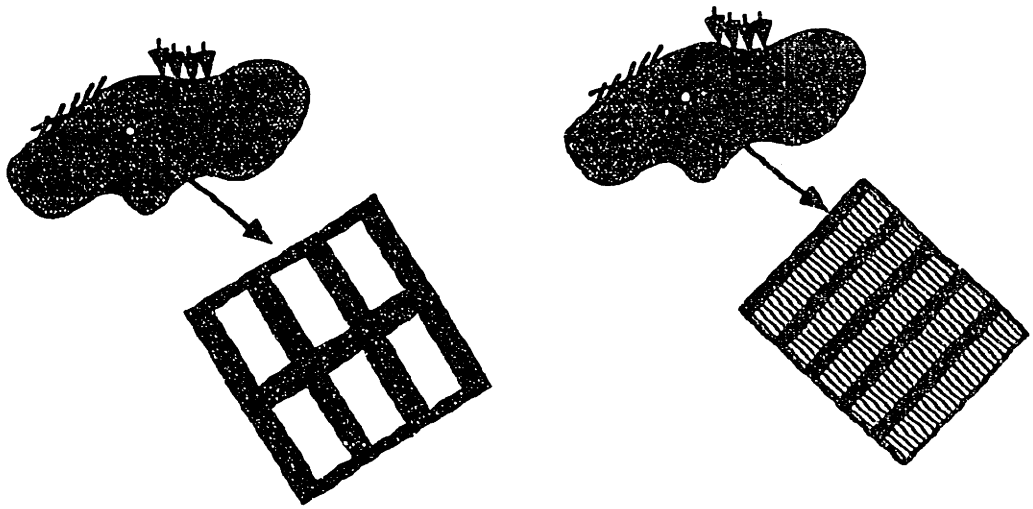


Figure 2.8. Rectangular hole and Rank-2 microstructures
(Adapted from Bendsoe_92b)

Figure 2.11 shows a comparison between these two microstructures. Figure 2.11(a) shows a microstructure consisting of unit cells with rectangular voids while figure 2.11(b) illustrates rank 2 microstructure. The rank 2 microstructure consists of layers of stiff material and rank 1 material. Rank 1 material in turn consists of layers of stiff and soft material. Allaire and Kohn [Allaire_92] have used rank 2 microstructure and have shown that for a composite material of fixed ratio of the two constituent materials, the rank 2 layering yields the stiffest composite. The bulk density is a function of the densities of the constituent materials and their ratio in the composite. The optimal shapes solved for using rank 2 layering do not have clearly defined holes. Instead, the structures obtained by this approach suggests that truely optimal structure may have continuously varying density due to continuously varying ratio of the two constituent materials. Such composite structures are difficult to manufacture since it is not economically possible to control the density and ratio of the constituents within a structure.

The geometry of the structure, obtained by the above methods, yields uneven boundaries, especially when the mesh is not very dense. Papalambros *et.al.* [Papalambros_90] use image processing techniques to extract a feasible initial shape from the topology designed using the homogenization method [Bendsoe_88]. They then use this shape as the starting shape for conventional shape optimization by boundary variation. A similar two stage process has been used by Bendsoe [Bendsoe_91] to obtain combined topology and shape optimization for 2D structures.

Binary Indicator Particles

Anagnostou *et. al.* [Anagnostou_92] use a combinatorial optimization procedure for shape optimization where they represent shapes using a lattice of binary indicator particles. The binary indicator particles are either on or off indicating the presence or absence of material. This kind of geometry representation is ideally suited for probabilistic optimization algorithms, since the variables are discrete. The optimal shape that minimizes a cost function is computed using simulated annealing. The method has been applied to several structural and heat transfer examples.

2.3.4. Design Objectives and Constraints

In this section, the most commonly used design criteria or functional requirements in structural synthesis are described. These functional requirements are stated in the form of design criteria and one such criteria is chosen as the design objective to be maximized or minimized. The actual choice of the objective function and constraints depends on the

designers intent and the application. Most of the criteria described here can be cast as the objective function or as a design constraint in the optimization problem statement.

In early applications of structural optimization, the goal was primarily to design light weight structures. The objective function was often therefore, chosen to be the weight or the volume of the structure. In the design of trusses, minimizing the weight of the structure subject to constraints on the plastic collapse load leads to a linear programming problem. Since very efficient and reliable linear programming algorithms exist, this formulation has been very popular in truss design. The plastic collapse constraints are formulated using limit analysis, a good explanation of which can be found in [Kirsch_81].

Another important design criterion is the stress and strain distribution in the structure under the applied loads. Often such criteria are stated in the form of design constraints where the stress or strain is not to exceed a certain design limit anywhere in the structure. These constraints are usually nonlinear. For some simple trusses and frames it is possible to derive analytical expressions for stress or strain as functions of the design variables. In general, however it is necessary to use numerical methods to compute the stresses and strains in the structure. These constraints are difficult to impose for 2D and 3D structures because the stresses and strains vary over the structural domain. It is not possible to predict the location where the stress would be maximum. Therefore, stress constraints have to be imposed at a large number of points in the structure resulting in a very large number of constraint equations. Furthermore, when numerical methods are used to compute the stresses, loads are often modeled as point loads and the stresses can be artificially large in the immediate neighborhood of the loads. These stresses then tend to be the active constraints dominating the design. The stress criterion in structural design is sometimes stated as the design objective by defining the objective as minimization of the maximum stress in the structure.

In the design of many planar and spatial structures, the primary design consideration is often to improve the stiffness to weight ratio. It is desirable to make the structure stiffer for a given weight of the structure, so that it deflects less under applied loads. This criteria can be stated as a constraint on the deflection of some point on the structure under the applied loads. Alternately, it can be stated as the design objective where one desires to minimize the deflection of a point on the structure.

The overall stiffness of the structure can also be maximized by minimizing the mean compliance of the structure or the total strain energy at equilibrium. In structural

optimization literature, mean compliance has been defined as twice the strain energy absorbed by the structure under the given loads [Haug_86, Haftka_92, Bendsoe_88]. Stiffer structures deflect less under applied loads and hence absorb less energy for a given load, therefore they have lower compliance. Maximizing the stiffness of the structure subject to constraints on the weight of the structure leads to designs that use material more efficiently and therefore have a better stiffness to weight ratio.

In the design of thin bars and membrane like structures, buckling is sometimes an important design criteria. For design of trusses and frames, the Euler buckling load is often modeled as a design constraint. Simulation of buckling for membranes is often non-trivial and therefore, expensive numerical methods have to be used to evaluate the buckling load. In the design of vibrating structures, the design criteria may be to maximize the natural frequency of the structure. Again for most types of structures, numerical methods are used to evaluate the natural frequency and their sensitivity with respect to the design variables. More details of such methods can be found in [Haftka_92].

2.4. Optimization algorithms

Many design or synthesis problems can be stated as an optimization problem where the design is parameterized and the optimal values of the design parameters are found by maximizing or minimizing an objective function subject to design constraints. Such optimization problems arise in a variety of design and decision making contexts and therefore, many general purpose numerical algorithms exist for solving them. A general optimization problem involves solving for the optimal values of a multi-variable function subject to a number of equality and inequality constraints. Algorithms for solving such problems are commonly referred to as mathematical programming algorithms, and the discipline that deals with such parameter optimization is often called mathematical programming. As mentioned in chapter 1, when the optimization problem involves only linear function we have linear programming problems and when some of the functions are nonlinear, the optimization problem is said to be a nonlinear programming.

A large number of algorithms have been proposed for all classes of linear and nonlinear programming. The best algorithm for solving a particular optimization problem often depends on the nature of the functions involved. It is often possible to design algorithms that are particularly suited for an application of interest. In this section, a brief historical perspective of algorithms used for structural optimization is given, and the algorithms are

broadly classified as the special purpose optimality criteria algorithms and the general purpose mathematical programming algorithms.

2.4.1. Historical Perspective

Mathematical programming is a well established discipline with diverse applications such as design optimization, decision making (operations research) in business and economics, applied mathematics, optimal controls and many more. Mathematical programming originated as a formal field of study in the operations research community where mathematical techniques were sought to help in business and economic analysis and decision making. However, many of the calculus based algorithms of optimization existed long before. Many general purpose optimization algorithms that have evolved from this branch of mathematics can be applied to structural optimization as well. The main difficulty in structural optimization is the high computational cost involved in evaluating functions that describe structural behavior. Typically each such function evaluation requires a structural analysis using numerical methods such as the finite element method. In addition most mathematical programming algorithms require sensitivity of the functions such as its gradient vector or hessian matrix. Due to the high cost of function evaluation and sensitivity analysis, algorithms that use few function evaluations and do not need the hessian matrix are preferred in structural optimization. A certain special class of algorithms have been proposed for structural optimization that make use of the understanding of the structural behavior and the nature of the optimal design. These algorithms are referred to as the optimality criteria methods. The optimization algorithms used for structural optimization can therefore be classified into optimality criteria methods and the mathematical programming algorithms as shown in figure 2.12.

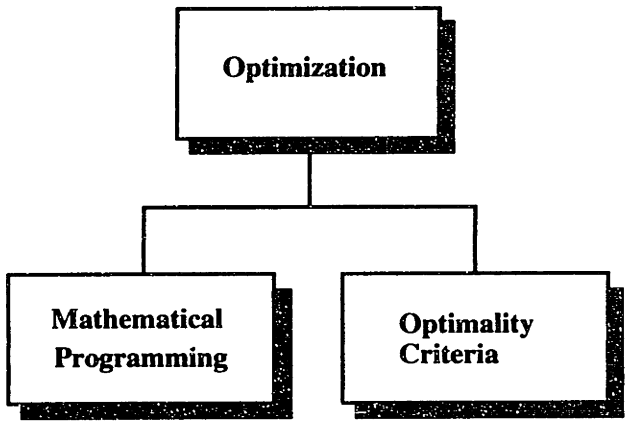


Figure 2.12. Classification of algorithms used for structural optimization

Optimality criteria methods initially evolved as a set of ad hoc or heuristic methods that directly solve the Kuhn Tucker optimality criteria (see Appendix A). Often such methods originated from intuitive reasoning about the nature of the optimal design such as the fully stressed structure. However, since these methods are non-rigorous and not easily generalizable they have not received full acceptance from the optimization / mathematical programming community. Mathematical programming algorithms on the other hand have been extensively studied and applied to a wide variety of applications. Optimality criteria methods tend to be special purpose and are fine tuned for a specific application and therefore they typically perform very efficiently for that application. The distinction between these two classes is fading since research in the past decade has established [Fleury_79, Fleury_86] that some of the dual methods of mathematical programming are very closely related to the optimality criteria methods. In fact some of the popular optimality criteria methods can be derived from these dual methods.

2.4.2. Optimality Criteria Methods

The optimal value of the variables satisfy a set of conditions referred to as the optimality criteria (also known as the Kuhn-Tucker conditions, see Appendix A). These are a set of necessary conditions that are satisfied by all the local maxima and minima of the optimization problem. If it can be shown that all the functions defining a nonlinear programming problem are convex then there exists only one unique optima which is the global minima of the optimization problem. Such a nonlinear program is called a convex program. For convex programs, one may solve the nonlinear equations of its optimality criteria using methods such as Newton's method to obtain the optimum values. However, in general, nonlinear programming problems commonly encountered in structural optimization have many local maxima and minima. Since the necessary conditions for optimality do not distinguish between local maxima and local minima, it is not possible to directly solve these equations for a minima. Moreover when there are many inequality constraints, its often not possible to predict which of these constraints would be active at the optimal solution. As a result, it is often not easy to solve the optimality conditions directly for a general nonlinear programming problem.

Optimality criteria methods use recurrence relations to update the variables until they satisfy the optimality conditions to a given tolerance [Haftka_92, Morris_82]. When there are many inequality constraints, it is assumed that the active constraints are known *a priori* or can be guessed. The recurrence relations are obtained from the optimality criteria using certain simplifying assumptions. A commonly used assumption is that the objective

function is nearly linear in the design variables whereas, the constraint functions are linear in the reciprocal variables. This is true for the truss optimization problems for which these methods were initially developed. Consider the following optimization problem expressed in terms of the reciprocal variables y,

$$\min\ f(y),\ \text{subject to}\ g_j(y) \geq 0,\ j = 1,..r_a$$

The optimality conditions for this problem may be simplified to the following form,

$$(x_i)^2 f_i - \sum_{j=1}^{r_a} \lambda_j c_{ij} = 0,\ \text{or}\ \ x_i = \left(\frac{1}{f_i} \sum_{j=1}^{r_a} \lambda_j c_{ij} \right)^{1/2} \qquad (2.4.1)$$

where, λ_j are the Lagrange multipliers, $f_i = \dfrac{\partial f}{\partial x_i}$ and $c_{ij} = -\dfrac{\partial g_j}{\partial y_i}$, i=1,...n. This relation may be converted into a recurrence relation in a number of different ways. For example, the exponential rule is based on rewriting equations (2.4.1) as,

$$x_i^{k+1} = x_i^k \left(\frac{1}{(x_i^k)^2 f_i} \sum_{j=1}^{r_a} \lambda_j c_{ij} \right)^{1/\eta},\ i=1,..n \qquad (2.4.2)$$

η gives control over the stepsize taken at each iteration. For larger values of η the change in the design variables is smaller per iteration.

Calculation of the Lagrange multipliers is made difficult when there are many inequality constraints because some of them may not be active at the optimal solution. If we assume that all the r_a constraints are active, then one can solve for the Lagrange multipliers by substituting the expression for the updated variables into the linearized form of the constraints. Optimality criteria methods are easy to apply to structural optimization problems where it is possible to predict the active constraints at the optimal solution.

2.4.3. Mathematical Programming Algorithms

The mathematical programming algorithms used in structural optimization may be broadly classified as calculus based and probabilistic search algorithms. Common examples of probabilistic search algorithms are simulated annealing and genetic algorithms. This classification is illustrated in figure 2.13. Calculus based algorithms can be classified further into those that directly solve for the optimal solution of the nonlinear program and those that iteratively construct approximate sub problems and solve for their optimal

solution at each iteration. The latter class of algorithms known also as sequential approximate optimization or 'approximation concepts' techniques have been particularly popular in structural optimization [Schmit_74, Fluery_79, Fluery_89]. The sub problems generated at each iteration are simple and easier to solve than the original problem.

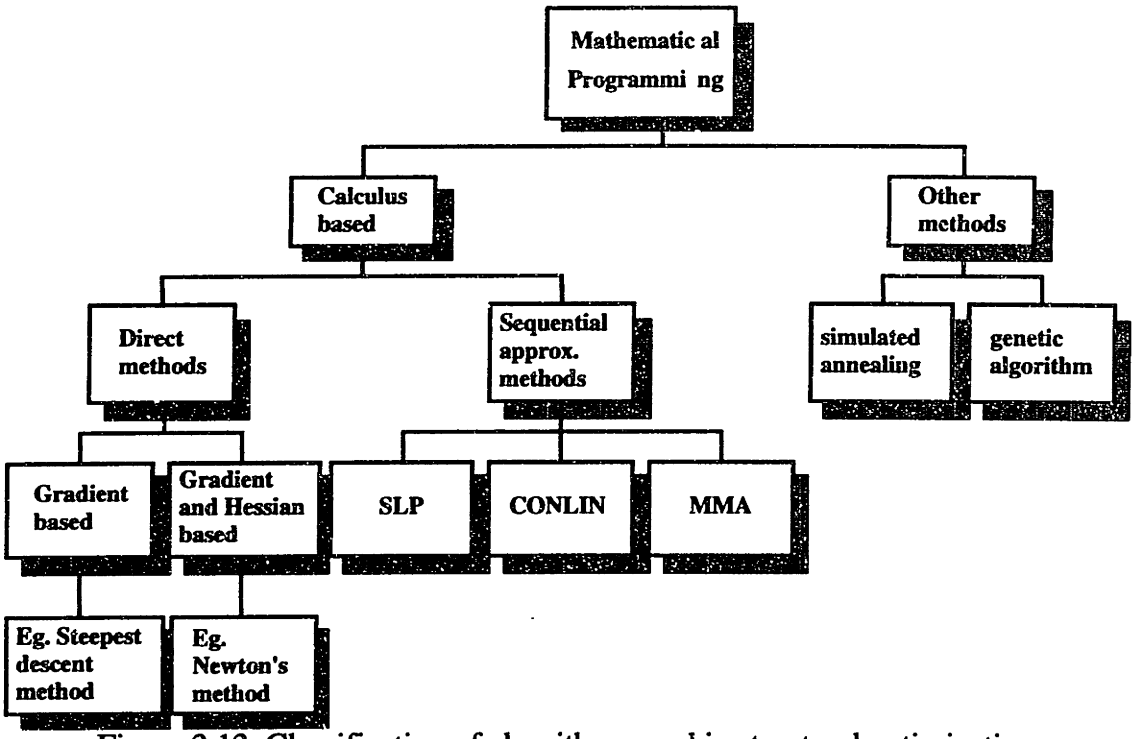


Figure 2.13. Classification of algorithms used in structural optimization

In the structural optimization context, it is convenient to classify the algorithms available to directly minimize a given nonlinear problem on the basis of the order of sensitivity information required by the algorithm. The gradient based algorithms require only first order sensitivity, that is the gradients of the functions. Examples of these are the steepest descent method, conjugate gradient methods, etc. Algorithms derived by modifying Newton's method on the other hand require the second order sensitivity, that is, the Hessian matrix as well. Details of such algorithms and applications for unconstrained and constrained optimization problems can be found in text books such as [Haftka_92] and [Morris_82]. A brief description of the key ideas behind various classes of algorithms is given below along with comments on their applicability for structural optimization.

Direct gradient based algorithms such as steepest descent for unconstrained optimization are inefficient for many applications, since they use local information only and move in the direction of steepest descent. The steepest descent direction is the direction opposite to that of the local gradient vector. When the objective function is badly scaled or

ill-conditioned with respect to the variables such algorithms perform poorly. An example of this is shown in figure 2.14 for a two dimensional objective function, where the steepest descent algorithm exhibits oscillatory behavior and converges very slowly. The figure shows constant value contours of the objective function in its 2D design space. The gradient is orthogonal to the constant contour curve. In this example, the steepest descent direction is nearly orthogonal to the direction that leads to the minimum.

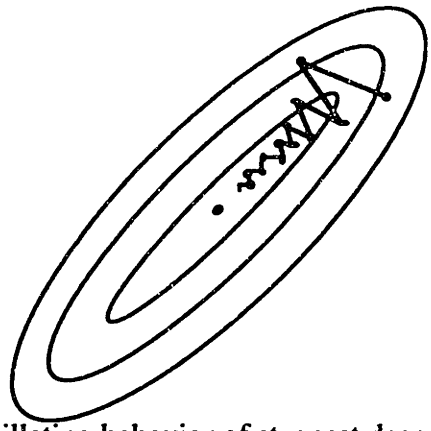


Figure 2.14. Oscillating behavior of steepest descent algorithm

Steepest descent methods do not take into account the local "curvature" of the objective function. Modified Newton algorithms use the hessian matrix and in effect construct a local quadratic approximation of the objective function and hence exhibit better convergence for ill-conditioned problems. However, second order sensitivity or the hessian matrix is in general expensive to compute. When the number of variables is large, the Hessian matrix is large and it requires considerable storage space and computing power to manipulate. This is particularly true in structural optimization where evaluating the sensitivity information of structural behavior often requires expensive numerical methods and the number of variables is usually very large. As a result even though theoretically modified Newton algorithms have a fast rate of convergence (fewer iterations), in practice they are not very efficient due to the high computational cost per iteration. In Newton methods, the descent direction ($\mathbf{d}^k$) for a function f(x) is computed by solving equations of the form

$$\nabla^2 f(\mathbf{x}^k)\mathbf{d}^k = -\nabla f(\mathbf{x}^k) \tag{2.4.3}$$

To ensure that the direction $\mathbf{d}^k$ is a descent direction, the hessian matrix $\nabla^2 f(\mathbf{x}^k)$ must be positive definite. However, this may not be the case for certain regions of the feasible domain. Therefore, in modified Newton methods positive constants are added to the diagonal elements of the hessian matrix, if necessary, to make it positive definite.

Quasi-Newton methods are among the most efficient methods for minimizing unconstrained optimization problems. Quasi-Newton methods require only the gradient, however, they construct an approximation to the inverse of the Hessian matrix using the gradients computed at each iteration. This approximation is updated at each iteration using a rank-one update matrix constructed using the gradient computed at that iteration. This approach therefore, does not require the evaluation of the hessian matrix at each iteration. Furthermore, by directly constructing the inverse of the approximate hessian matrix it reduces the computation associated with solving for the descent direction using equations of the form (2.4.3).

Many techniques exists for extending unconstrained optimization algorithms to handle constraints. When the constraints are linear, gradient projection methods can be used to enforce the constraints. This method consists of using a gradient based algorithm to find a descent direction and stepsize to update the current variables and then projecting it back to the feasible set. Let S be the set of feasible points. The variable is updated each iteration as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k(\bar{\mathbf{x}}^k - \mathbf{x}^k) \tag{2.4.4}$$

where, $\alpha^k \in [0,1]$ is a stepsize, $\bar{\mathbf{x}}^k$ is obtained by projecting on S the point obtained by the gradient method $(\mathbf{x}^k - s^k\nabla f(\mathbf{x}^k))$ where s^k is the stepsize computed by line minimization along the descent direction. It is assumed that S is a convex set so that the projected point is unique.

Penalty methods are applicable to general nonlinear programming problems. These methods convert constrained optimization problems into unconstrained optimization by constructing a new objective function which consists of a sum of the original objective function and penalty terms for constraint violation. The simplest among these is the quadratic penalty method that adds a square of the constraint functions. The penalized objective function is of the form

$$L_c(\mathbf{x}) = f(\mathbf{x}) + \frac{c}{2}\|\mathbf{h}(\mathbf{x})\|^2 \tag{2.4.5}$$

For large values of c, there is a high cost of infeasibility so that the unconstrained minimum of this function is nearly feasible and would approach the constrained minimum of $f(\mathbf{x})$ subject to the constraints $\mathbf{h}(\mathbf{x})$, in the limit as $c \to \infty$.

Multiplier methods also penalize constraint violation. The constrained optimum is found by the unconstrained optimization of the Augmented Lagrange function, defined as,

$$L_c(\mathbf{x},\lambda) = f(\mathbf{x}) + \lambda^t \mathbf{h}(\mathbf{x}) + \frac{c}{2}\|\mathbf{h}(\mathbf{x})\|^2 \tag{2.4.6}$$

If λ is equal to the Lagrange multipliers λ^* satisfying the Kuhn-Tucker optimality conditions, then for a sufficiently large c, the unconstrained local minima of $L_c(\mathbf{x},\lambda^*)$ is the constrained local minima of $f(\mathbf{x})$ subject to the constraints $\mathbf{h}(\mathbf{x})$. In the method of multipliers, λ is updated each iteration as

$$\lambda^{k+1} = \lambda^k + c^k \mathbf{h}(\mathbf{x}^k) \tag{2.4.7}$$

where, $\mathbf{x}^k$ is the unconstrained minimum of $L_{c_k}(\mathbf{x},\lambda^k)$. It can be shown that the sequence $\{\mathbf{x}^k\}$ converges to a constrained local minima of $f(\mathbf{x})$, while $\{\lambda^k\}$ converges to the Lagrange multipliers λ^*.

Sequential optimization algorithms, construct a sequence of sub problems such that the solutions to these sub problems tend towards the solution of the original problem. Sequential linear programming (SLP) and sequential quadratic programming (SQP) are some of the earliest sequential optimization algorithms. Sequential optimization algorithms have been of considerable interest to structural optimization research, since they often help to reduce the number of function and gradient evaluations. Therefore, a number of sequential approximate algorithms have been proposed specifically for this application. In structural optimization, evaluation of the objective and constraint functions and their gradients is very expensive. The number of variables is also typically very large. Due to these reasons "some approximations concepts" were first proposed for structural optimization by Schmit and Farshi [Schmit_74], to reduce the number of function and gradient evaluations required during optimization. Commonly used approximations include linear approximation, leading to SLP, linearization in reciprocal variables as well as hybrids of these two approximations. Quadratic approximations are usually avoided in structural optimization due to the high cost of constructing the hessian matrix. Details of some of the sequential approximate optimization algorithms that have been proposed for structural optimization and their evolution has been described in section 4.2 (Chapter 4).

The methods described so far typically converge to a local minima of the optimization. Except for convex optimization problems there is no guarantee that the minimum obtained is a global minimum. Probabilistic search methods search a larger space of the design domain and hence tend to identify the global optimum. These methods are in general very computationally expensive, but they are well suited for parallel computing. The two most popular probabilistic search algorithms are simulated annealing and genetic algorithms.

3 Shape and Topology Optimization

3.1. Overview

Traditional geometric modeling techniques have assumed that geometry is completely defined and modified by human input. Such tools have therefore emphasized ease of geometric input and proper user-interface for geometry editing by the user. However, in structural optimization, geometry is modified by an optimization algorithm. The conventional geometric modeling techniques are not ideally suited for this application.

In the previous chapter, an overview of the types of geometry representation and design variables used in structural optimization was described. Sizing variables do not change the overall shape of the structure, instead they change the cross-section of bars/beams or vary the thickness of plates etc. Shape variables are parameters that describe the boundaries of the shape and therefore varying them, varies the boundary of the shape but not its topology. To enable shape and topology optimization we need shape representations and design variables that allow both the shape and topology to vary. Topology optimization by homogenization method assumes that the material is porous and solves for the optimal material distribution. The optimal solution assigns the optimal porosity for each element in the model such that elements in regions where material is not needed are highly porous and are removed to create holes.

The shape is represented using a "shape density" function in this thesis such that the boundaries of the shape are contours of this function corresponding to a threshold value of the density. Section 3.2 describes shape representation using the shape density function and the design variables associated with it. The objective function and the constraints that are used to define the structural optimization problem are described in section 3.3. Approximate relations are used to characterize the variation of material properties with the shape density function. The relations that were studied in this research are described in section 3.4. The shape is modified by an optimization algorithm and the structural

properties are evaluated at each iteration using the finite element method. Section 3.5 describes how the finite element method is used to analyze the structural behavior taking into account that the density function and the material properties vary continuously over the domain. The nonlinear programming algorithm used to evaluate the optimal design requires first order sensitivity information. Section 3.6 describes how the gradient of the objective function and the constraint are evaluated.

3.2. Shape density function representation

To model geometry as a variable, we use the contours of a function that we will refer to here as the shape density function (or simply as density function). Contours of this function corresponding to a threshold value are defined as the boundaries of the shape so that regions where the value of the function is below the threshold are not part of the geometry. Hence, the shape may be defined using the following implicit expression

$$\phi(x,y) \geq \phi_{th} \tag{3.2.1}$$

where, $\phi(x,y)$ is the density function and ϕ_{th} is the threshold value of the density function. The equality in the above expression corresponds to the boundary of the shape.

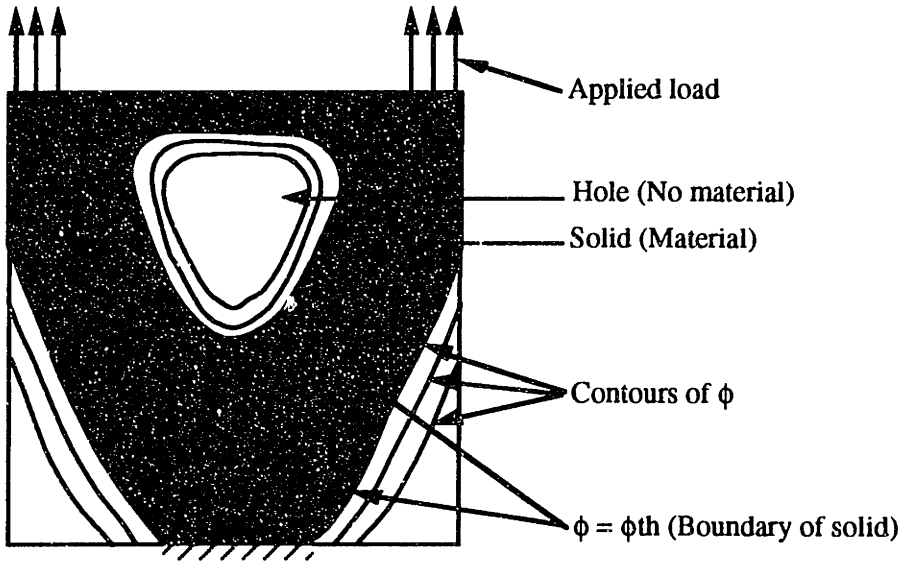


Figure 3.1. Shape representation using shape density function

Figure 3.1 illustrates this shape representation. The density function is defined within the rectangular feasible region. Contours of the density function corresponding to constant values of density are plotted. The shaded region represents the interior of a 2D shape and the contour of the density function corresponding to the threshold value ϕ_{th} is the boundary

of the solid. In this example, there are two contours corresponding to the threshold value and hence there are two boundary curves, one internal and one external boundary. Shape representation using a contour of the shape density function enables the entire geometry to be treated as a variable. By changing the shape density function it is possible to not only modify existing boundaries but also to create new internal boundaries.

The density function takes values ranging from the threshold value, ϕ_{th}, to 1. It takes value 1 where the material is fully dense and the threshold value ϕ_{th} where there is no material. It can also take intermediate values. A new internal boundary corresponding to a hole, for example, would be created if the value of the density function decreases to the threshold value in a region.

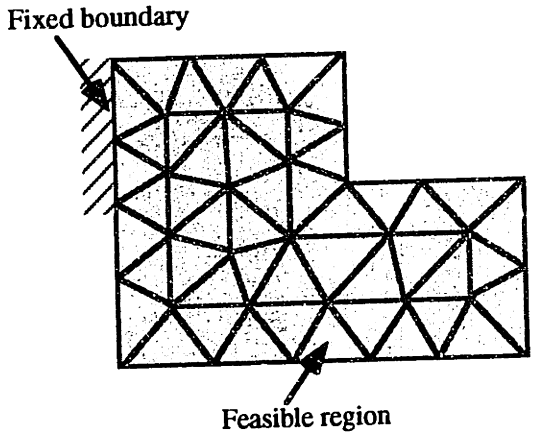


Figure 3.2. Triangulated feasible region

To define the shape density function, the feasible domain is divided into triangular finite elements as shown in figure 3.2. The density function is linearly interpolated over each element using the same interpolation functions that were used to represent the displacement fields (equation 2.2.7). Within each element the density function is given by,

$$\phi = \phi_1 L_1 + \phi_2 L_2 + \phi_3 L_3 \tag{3.2.2}$$

where, ϕ_i are the nodal values of the density function and L_i are the linear interpolation functions described in section 2.2.

$$L_i = (a_i + b_i x + c_i y)/2\Delta \tag{3.2.3}$$

$$a_i = x_j y_k - x_k y_j, \tag{3.2.4}$$

$$b_i = y_j - y_k \tag{3.2.5}$$

$$c_i = x_k - x_j \tag{3.2.6}$$

where, i, j and k are cyclically allotted the values 1, 2 and 3. Δ is the area of the triangular element.

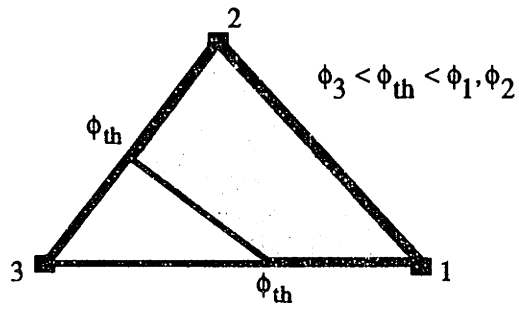


Figure 3.3. Triangular element

(x_i, y_i) are the nodal coordinates of the nodes of the element. These simple interpolation functions L_i, often referred to as the area coordinates, ensure a linear interpolation of nodal densities within the element. A contour of the density function corresponding to the threshold value passes through the element if some nodes of the element have nodal density values higher than the threshold value while others have values below. Since we have used linear interpolation, the contour within an element is linear. This contour can be obtained by joining the points on the edges of the triangle that have density value equal to the threshold value as shown in figure 3.3. A C^0 continuous shape density function ensures C^0 continuous boundaries for the final shape, providing a means of combined shape and topology optimization.

The mesh used for defining the density function can also serve as the finite elements for structural analysis. The values of the shape density function at the nodes serve as the design variables of the optimization problem. Initially, the nodal shape density values are set equal to unity at each node so that the geometry is identical to the feasible region. During the optimization process the nodal density values, ϕ_i are modified by the optimization algorithm which iteratively searches for the optimal values of the nodal densities such that the overall stiffness of the structure is maximized. The objective function is described mathematically in the next section. The optimization algorithm used in this thesis is described is chapter 4.

3.3. Optimization objective and constraints

In this thesis, we will solve for the optimal geometry of structural components with the objective of maximizing stiffness, while setting a limit on weight and avoiding interference with other components. We limit the study to planar (2D) problems, such as plane stress

and plane strain. To maximize the stiffness of a component, we minimize the strain energy of the structure at equilibrium.

Structural synthesis is the inverse of the structural analysis problem. The structural analysis problem is typically stated as a principle of virtual work (PVW), (see section 2.2.1).

$$\int_{\Omega} \{\delta\varepsilon\}^t [\mathbf{D}] \{\varepsilon\} \, d\Omega = L(\delta\mathbf{u}) \tag{3.3.1}$$

$$L(\delta\mathbf{u}) = \int_{\Omega} \mathbf{f} \cdot \delta\mathbf{u} \, d\Omega + \int_{\Gamma_t} \mathbf{t} \cdot \delta\mathbf{u} \, d\Gamma \tag{3.3.2}$$

where $\{\varepsilon\}$ is the strain vector, $[\mathbf{D}]$ is the elasticity matrix and $L(\delta\mathbf{u})$ is the virtual work done by the applied body force $\mathbf{f}$ and traction $\mathbf{t}$. $\delta\mathbf{u}$ is the virtual displacement and Ω is the domain over which the PVW applies. The domain Ω represents the shape and topology of the component whose structural properties are being analyzed. We use the finite element method to solve for the displacement field $\mathbf{u}(\mathbf{x})$ for $\mathbf{x}$ belonging to Ω.

The shape and topology synthesis problem involves solving for a domain Ω that optimizes some structural property for given loading and boundary conditions. The main challenge is in representing the domain Ω as a variable. As explained in the previous section, we define Ω as the region within the original feasible domain Ω_0 where the shape density function has a value greater than the threshold value. The design problem specification consists of defining a feasible domain, the applied loads and the boundaries that are fixed, as shown in figure 3.2.

The structural optimization problem is stated as, the minimization of twice the strain energy of the structure (or mean compliance) $L(\mathbf{u})$ subject to a constraint on weight. The minimization problem may then be written as,

Minimize $L(\mathbf{u})$

$$L(\mathbf{u}(\phi)) = \int_{\Omega} \mathbf{f} \cdot \mathbf{u}(\phi) d\Omega + \int_{\Gamma_t} \mathbf{t} \cdot \mathbf{u}(\phi) d\Gamma \tag{3.3.3}$$

subject to,

$$W(\phi) = \int_{\Omega} \phi \, d\Omega \leq W_0 \tag{3.3.4}$$

$$\int_{\Omega_o} \{\delta\varepsilon\}^t [\mathbf{D}(\phi)] \{\varepsilon\} \, d\Omega_o = L(\delta\mathbf{u}) \tag{3.3.5}$$

where, $\phi(\mathbf{x})$ is the shape density function. Equation (3.3.4) describes the constraint that the weight (volume) W of the component should be less than or equal to W_O. $L(\mathbf{u})$ is twice the work done by the applied forces due to the displacement $\mathbf{u}$ (often referred to as the mean compliance in structural optimization literature). Note that $L(\mathbf{u})$ is also twice the strain energy of the structure at equilibrium under the applied loads. In this thesis, we will refer to $L(\mathbf{u})$ as the strain energy or the mean compliance. Larger compliance implies an ability to absorb more energy for a given load. Stiffer structures deflect less and hence have a lower compliance since they absorb less energy for a given load. Therefore, minimizing the mean compliance (or the strain energy of the structure) is equivalent to maximizing the stiffness. The design objective is to find the shape that provides maximum stiffness for a given weight and fits in the feasible domain.

3.4. Material property variation with density

When shape defined by the density function varies the structural properties must vary accordingly. This implies that the material property coefficients defined in the matrix [D] equation (2.2.4a) must depend on the density function $\phi(x,y)$. In chapter 2, section 2.3.3 the homogenization based methods of topology optimization was discussed. In this approach the material was assumed to be porous or a composite of 2 materials. The optimal topology was obtained by computing the optimal distribution of porosity or ratio between the constituents of a composite. Homogenization method is used to predict how the material property coefficients vary as a function of porosity (void fraction) or as a function of ratio of the constituents when the material is assumed to be a composite. The material property - density relations obtained using homogenization technique depends on the microstructure assumed for the porous material or the compcsite. The optimal structures obtained also differ based on the microstructure assumed. Bendsoe and Kikuchi [Bensoe_88] have assumed a microstructure consisting of rectangular voids in each unit cell (see section 2.3.3). This assumption leads to highly nonlinear material property density relations that are difficult to integrate over each element. Therefore, they make the assumption that each element has a constant porosity. The optimal shapes predicted by their approach has clearly defined holes and very little region having intermediate porosity, that is the material is either fully dense (no porosity) or zero density.

Allaire and Kohn [Allaire_92] have used rank 2 microstructure (see section 2.3.3). This assumption leads to analytical relation between material properties and densities, however, the shapes obtained by this approach do not have clearly defined topology. The ratio between the constituents of the composite varies gradually with no clearly defined boundaries between holes and material.

In this section, we discuss the material property - density relations assumed in this thesis. The goal is to seek relations that simple and therefore easy to integrate over each element when density varies linearly with each element. Also we are interested in relations that lead to clearly defined topologies so that the final shape obtained is fully dense and the density function transitions sharply at the boundary from full density to the lowest possible (threshold value). In the subsections below we discuss the material property density relation studied in this thesis.

3.4.1. Linear approximate relation

When the shape of the component changes, its structural properties change. The structural properties (stiffness, in particular) must depend on the shape density function since we represent shape using a contour of this function. This relationship between structural properties and the density function is approximated by assuming that the material properties vary linearly with the density. This assumption may be intuitively explained by noting that if the density decreased in a region, the material property coefficients would decrease by our assumption causing the material to become weaker in that region. The optimization algorithm therefore, decreases the density in regions where material is under-utilized causing either new boundaries (holes) to be created in such regions or causing existing boundaries to shrink inwards. In equation (3.3.3), therefore, the strain energy of the structure or $L(\mathbf{u})$ is expressed as a function of the shape density function ϕ. In this section we describe a linear approximate relation between material properties and density that is valid for small variations of the density function. Assuming linear elastic material, the stress-strain relation for a two-dimensional plane stress case may be written as :

$$\{\sigma\} = [\mathbf{D}_1(\phi)]\{\varepsilon\} \tag{3.4.1}$$

$$[\mathbf{D}_1(\phi)] = \begin{bmatrix} d_{11} & d_{12} & 0 \\ d_{12} & d_{11} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \tag{3.4.2}$$

$[\mathbf{D}_l]$ is the elasticity matrix with linear approximate dependence on the density function. Assuming that the elasticity modulus is linearly dependent on the density and linearizing the resultant coefficients d_{ij} about $\phi=1$, we get the following relations for the elasticity coefficients that are linear in ϕ.

$$
\begin{aligned}
d_{11} &= \frac{E}{1-\nu^2} + \frac{E(1+\nu^2)}{(1-\nu^2)^2}(\phi-1) \\
d_{12} &= \frac{E\nu}{1-\nu^2} + \frac{2E\nu}{(1-\nu^2)^2}(\phi-1) \\
d_{33} &= \frac{E}{2(1+\nu)} + \frac{E}{2(1+\nu)^2}(\phi-1)
\end{aligned}
\qquad (3.4.3a)
$$

where, E = Young's modulus of elasticity and ν = Poisson's ratio. At $\phi=1$, the second term in equations (3.4.3) vanishes, reducing d_{ij} to the usual elasticity coefficients of the plane strain-plane stress elasticity matrix. These coefficients are plotted as a function of the density function ϕ in figure 3.4. The curves represent the variation of the coefficients d_{ij} with the density function, when both the elastic modulus and Poisson's ratio are assumed to vary linearly with the density function. The tangent to these curves at $\phi=1$ represents the relation given by equations (3.4.3). As can be seen from the figure, the approximation is good only in the neighborhood of $\phi=1$. Some coefficients even become negative for small values of the density function. Therefore, the threshold value are set close to 1 so that the density function can vary only slightly within the geometry. The lower bound on the density function is set equal to the threshold value. In addition the use of threshold values close to 1 ensures that in the final optimal geometry there is a sharp transition of density values at the boundary from the maximum allowed value to the lowest possible value (below threshold value). Sharp transition of the density function at the boundaries lead to clearly defined boundaries and fully dense material in the interior of the shape.

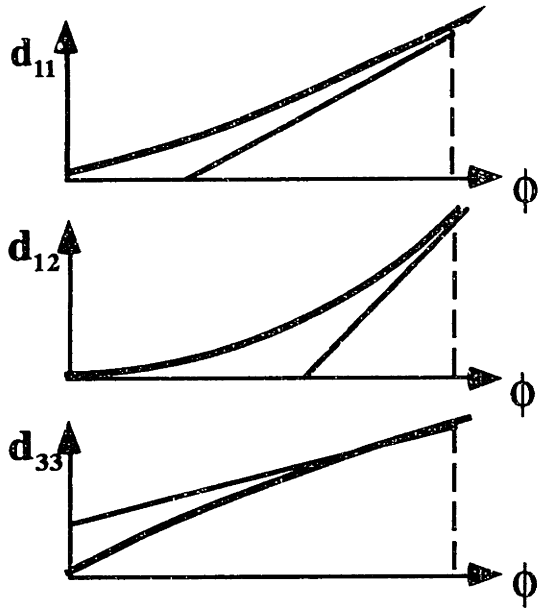


Figure 3.4. Material property coefficients versus density function

It is convenient to decompose the elasticity matrix into a constant component and one that depends linearly on the density function as

$$[D_1] = [D] + [dD](\phi - 1) \qquad (3.4.3b)$$

where [D] is the elasticity matrix defined in equation (2.2.4), while [dD] is made up of the coefficients of (ϕ-1) in equation (3.4.3). The details of the implementation of these material property relations in an optimization software is described in chapter 5.

3.4.2. Quadratic and higher order approximations

Rather than use linear approximations such as those explained in the previous section, it is possible to obtain a quadratic approximation by assuming that the elastic modulus of the material varies quadratically with respect to the density function. Similarly, one could make higher order approximations. Just as different microstructure assumptions lead to different material property-porosity relation when homogenization method is used, different assumptions on the variation of elastic modulus and poisson's ratio with density lead to different material-property density relation. The criteria in our case for choosing between the different relations is the sharpness of the density transition at the boundary. In other words, we want the material inside the shape to be fully dense (ϕ=1) and the material to have the lowest possible density where the holes are located. At the boundary we want the density to transition sharply from the highest value to the lowest value, so that we have clear and well defined boundaries. When the material is assumed to have a rank 2

microstructure the material tends to have gradual transition of density so that the bulk of the solid is porous. Similarly, when a linear relation is assumed we fail to get sharp boundaries. Since in this thesis we are interested in the design of structures of non-porous materials, we prefer a relation that leads to clearly defined boundaries with fully dense material in the interior. It was found that in general, higher order approximations of the material property-density relations lead to desired behavior.

Assuming p^{th} order variation of the elasticity modulus with the density function, we get the following material property-density variation,

$$d_{11} = \frac{E\phi^p}{1-\nu^2}$$
$$d_{12} = \frac{E\nu\phi^p}{1-\nu^2} \qquad (3.4.4)$$
$$d_{33} = \frac{E\phi^p}{2(1+\nu)}$$

Again the coefficients d_{ij} increase with the density function ϕ. In addition, for this approximation the material coefficients reduce to zero as density goes to zero, that is, $d_{ij}=0$ for $\phi=0$. Therefore, the threshold value on density ϕth can be set to be very small or nearly zero. The elasticity matrix can then be conveniently defined as

$$[D_p(\phi)] = [D]\phi^p \qquad (3.4.5)$$

where [D] is the elasticity matrix for plane stress-plane strain defined in equation (2.2.4).

When higher order material property-density relations are used the threshold value can be set to zero since the material coefficients do not become negative in the interval ($0 \le \phi \le 1$). The advantage of setting the threshold value to zero is that during the optimization process when the density is reduced to zero in certain regions the contribution of these regions to stiffness is also reduced to zero. This is equivalent to physically removing the finite elements in the region to create holes. However, when the threshold values are set to zero the optimal solutions obtained often contains regions where the density function has intermediate values, that is values that are neither zero or one. The transition of the density function values in the optimal solution is often very gradual so that the large portion of the shape have intermediate densities. While the truly optimal shapes may be composites whose density varies in the manner suggested by the optimal solution, it is often desirable

from manufacturing considerations to obtain solution that have fully dense solids. To achieve this we add a penalty function to our objective function that penalizes intermediate densities. The penalized objective function may be stated as

$$L_p(\mathbf{u}(\phi)) = \int_{\Omega} \mathbf{f}\cdot\mathbf{u}(\phi)d\Omega + \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{u}(\phi)d\Gamma + c_p\int_{\Omega}\phi(1-\phi)d\Omega \qquad (3.4.6)$$

In the above function the last term represents a penalty on intermediate densities. c_p is a penalty constant whose value is increased in steps in our implementation as described in chapter 5. As can be easily verified this term has the maximum value for ϕ=0.5 and has a minimum value at the extreme value of the interval [0,1]. A similar penalty function has been used by Allaire and Kohn [Allaire_92].

3.5. Implementation using Finite Element Method

The shape density function is assumed to be uniformly varying over the domain and is represented, as described in section 3.2, by a piece-wise linear interpolation. Since material properties depend on the density function, they too vary uniformly over the domain. The elasticity matrix, therefore, is a function of the density function by the approximate relations described in the previous section. In constructing the stiffness matrix, we integrate the principle of virtual work (equation 2.2.1), over each element to obtain the element stiffness matrix and then assemble these together to obtain the global stiffness matrix. We have used constant strain triangular finite elements for plane stress analysis. The displacement fields are therefore, represented by piece-wise linear interpolation. This representation when used in equation (2.2.5), yields the following linear displacement-strain relation for the element,

$$\{\varepsilon^e\} = [B]\{u_h^e\} \qquad (3.5.1)$$

Substituting this relation in the expression for the virtual internal energy (or the l.h.s of the principle of virtual work), we get, for each element

$$L^e(\delta\mathbf{u}) = \int_{\Omega_e}\{\delta\varepsilon^e\}^t[\mathbf{D}(\phi)]\{\varepsilon^e\}\,d\Omega_e = \{\delta u_h^e\}^t\left[\int_{\Omega_e}[B]^t[\mathbf{D}(\phi)][B]\,d\Omega_e\right]\{u_h^e\} \qquad (3.5.2)$$

$L^e(\delta\mathbf{u}) = \{\delta u_h^e\}^t[K_e]\{u_h^e\}$, where, $[K_e]$ is the element stiffness matrix,

$$[K_e] = \int_{\Omega_e}[B]^t[\mathbf{D}(\phi)][B]\,d\Omega_e \qquad (3.5.3)$$

Since we have assumed a linear interpolation for the displacement within each element, the matrix [B] is constant, resulting in a constant strain over the element. The elasticity matrix, however, depends on the density function. Depending on the type of relation assumed between the material properties and the density function, we need to integrate various powers of the density function over the element. In the linear approximate relation, using equation (3.4.3), we express the element stiffness matrix as

$$[K_e] = [B]^t[D][B]\Delta + [B]^t[dD][B]\Delta_\phi \tag{3.5.4}$$

where, Δ is the area of the triangle, [D] is the elasticity matrix for fully dense plane stress element and

$$\Delta_\phi = \int_{\Omega_e} (\phi - 1)d\Omega_e = \left(\frac{\phi_i + \phi_j + \phi_k}{3} - 1\right)\Delta \tag{3.5.5}$$

and i, j and k are the nodes of the element e. Linear interpolation of the density function over the elements makes this integration relatively easy. For quadratic and higher order approximations, we need to integrate higher powers of the density function over the element. The following relation can be derived for a p^{th} order material property-density relation,

$$I^p = \int_{\Omega_e} \phi^p d\Omega_e = \int_0^1 \int_0^{1-L_1} (L_1\phi_i + L_2\phi_j + L_3\phi_k)^p 2\Delta dL_2 dL_1 \tag{3.5.6}$$

where, $L_1 + L_2 + L_3 = 1$

$$I^p = 2\Delta \frac{\phi_i^{p+2}(\phi_k - \phi_j) + \phi_j^{p+2}(\phi_i - \phi_k) + \phi_k^{p+2}(\phi_j - \phi_i)}{(p+1)(p+2)(\phi_i - \phi_j)(\phi_k - \phi_j)(\phi_i - \phi_k)} \tag{3.5.7}$$

However, in practice it is not possible to use this relation directly since the denominator can become zero if the density function values of any two nodes are equal. The denominator can be factored out of the numerator once the value of 'p' is specified using symbolic manipulation software. For example for p=4, we can express I^p as,

$$\begin{aligned} I^{(4)} = \frac{\Delta}{15}(&\phi_i^4 + \phi_j^4 + \phi_k^4 + \phi_i^3\phi_j + \phi_i^3\phi_k + \phi_j^3\phi_i + \phi_j^3\phi_k \\ &+ \phi_k^3\phi_i + \phi_k^3\phi_j + \phi_i^2\phi_j^2 + \phi_j^2\phi_k^2 + \phi_k^2\phi_i^2 \\ &+ \phi_i^2\phi_j\phi_k + \phi_j^2\phi_i\phi_k + \phi_k^2\phi_j\phi_i) \end{aligned} \tag{3.5.8}$$

These relations for I^p can be used to express the element stiffness matrix for higher order material property-density relations as,

$$[K_e] = [B]^t[D][B]\, I^p \tag{3.5.9}$$

These element stiffness matrices can then be assembled into a global stiffness matrix for finite element analysis (equation 2.2.6).

3.6. Sensitivity evaluation

As explained in chapter 2, most mathematical programming algorithms use sensitivity information such as the gradient or the hessian of the objective function and constraints with respect to the design variables. In this section, we describe the method for finding the gradient of $L(\mathbf{u})$ with respect to the density function. One could use adjoint variable technique described in section 2.2.2 to derive the gradient. However, here we derive the same result by directly differentiating the equilibrium equations expressed in the form of linear simultaneous equations using the finite element method (equation 2.2.6). We restate the equations here for convenience,

$$[\mathbf{K}]\{\mathbf{u}_h\} = \{\mathbf{F}\} \tag{3.6.1}$$

where $[\mathbf{K}]$ is the global stiffness matrix, $\{\mathbf{u}_h\}$ is the displacement vector, that is, a discrete representation of the displacement field, and $\{\mathbf{F}\}$ is the resultant external applied load vector. In terms of these quantities, the compliance in equation (3.3.3) may be written as,

$$L(\mathbf{u}) = \{\mathbf{u}_h\}^t[\mathbf{K}]\{\mathbf{u}_h\} \tag{3.6.2}$$

Taking the partial derivative of equation (3.6.2) with respect to the nodal values of the density functions yields,

$$\frac{\partial}{\partial \phi_i} L(\mathbf{u}) = \{\mathbf{u}_h\}^t \frac{\partial[\mathbf{K}]}{\partial \phi_i}\{\mathbf{u}_h\} + \frac{\partial\{\mathbf{u}_h\}^t}{\partial \phi_i}[\mathbf{K}]\{\mathbf{u}_h\} + \{\mathbf{u}_h\}^t[\mathbf{K}]\frac{\partial\{\mathbf{u}_h\}}{\partial \phi_i} \tag{3.6.3}$$

The derivative of the displacement field with respect to the density function can be obtained by differentiating equation (3.6.1), which yields,

$$\frac{\partial\{\mathbf{u}_h\}}{\partial \phi_i} = -[\mathbf{K}]^{-1}\frac{\partial[\mathbf{K}]}{\partial \phi_i}\{\mathbf{u}_h\} \tag{3.6.4}$$

Substituting this expression in equation (3.6.3), we get,

$$\frac{\partial}{\partial \phi_i} L(\mathbf{u}) = -\{\mathbf{u}_h\}^t \frac{\partial [\mathbf{K}]}{\partial \phi_i} \{\mathbf{u}_h\} \tag{3.6.5}$$

The derivative of $L(\mathbf{u})$ with respect to density is negative if $\partial[\mathbf{K}]/\partial\phi_i$ is positive definite. A negative value of the derivative implies that when the density function increases the compliance of the structure decreases while its stiffness increases. Since this is the desired behavior we choose material property-density relations such that $\partial[\mathbf{K}]/\partial\phi_i$ is positive definite. In using equation (3.6.5), it is not necessary to construct the derivative of the global stiffness matrix with respect to each nodal density value. In fact, the element stiffness matrix of only those elements that contain the node are functions of its nodal density value. Therefore, the partial derivative with respect to a nodal density can be constructed as

$$\frac{\partial}{\partial \phi_i} L(\mathbf{u}) = -\sum_e \{\mathbf{u}_h^e\}^t \frac{\partial [\mathbf{K}_e]}{\partial \phi_i} \{\mathbf{u}_h^e\} \tag{3.6.6}$$

where the summation is carried out over the elements that contain the i^{th} node, $[K_e]$ is the element stiffness matrix and $\{\mathbf{u}_h^e\}$ is the displacement vector corresponding to the element.

The derivative of the element stiffness matrix with respect to a nodal density value can be computed by differentiating equation (3.5.3). If a linear approximate material property-density relation is assumed, then differentiating equation (3.5.4), we get,

$$\frac{\partial [K_e]}{\partial \phi_i} = \frac{\Delta}{3} [B]^t [dD][B] \tag{3.6.7}$$

Differentiating equation (3.5.9), we can get the following expression for higher order material property - density relations,

$$\frac{\partial [\mathbf{K}_e]}{\partial \phi_i} = [B]^t [D][B] \frac{\partial I^p}{\partial \phi_i} \tag{3.6.8}$$

For example, when p=4, differentiating equation (3.5.8), we get,

$$\frac{\partial I^{(4)}}{\partial \phi_i} = \frac{2}{15}\phi_i^3 + \frac{1}{30}(\phi_j^3 + \phi_k^3) + \frac{1}{10}(\phi_i^2\phi_j + \phi_i^2\phi_k) + \frac{1}{15}(\phi_j^2\phi_i + \phi_k^2\phi_i) + \frac{1}{30}(\phi_j^2\phi_k + \phi_k^2\phi_j) + \frac{1}{15}\phi_i\phi_j\phi_k \tag{3.6.9}$$

The nonlinear programming algorithm used in this thesis requires only the gradient of the functions with respect to the design variables. The methods described in this section can be extended to derive the hessian matrix of the compliance of the structure if needed. The constraint function on the weight / volume of the structure is the integral of the density function over the feasible domain. Its gradient is trivial to compute and the element by element method described in this section was used to evaluate it.

4 A Sequential Approximate Optimization Technique

4.1. Overview

Engineering design and structural optimization problems are often stated as non-linear programming problems as illustrated in the earlier chapters. Typically these problems have a large number of variables and the evaluation of the objective function and the constraints involve computationally expensive analysis. In section 4 of Chapter 2, an introduction to some of the commonly used nonlinear programming algorithms is presented. A large number of algorithms exist for nonlinear programming. None of the algorithms perform uniformly well for all kinds of problems. Most have been specifically designed to perform well for a certain class of problems.

Sequential approximate optimization techniques (or "approximation concepts" approaches) create a sequence of sub-problems that are easier to solve than the original problem. The approximate objective functions and constraints of the sub-problem are much cheaper to evaluate and the sub-problems are relatively inexpensive to solve. The function and gradient evaluations of the original optimization problem are typically performed only once per sub-problem. Unlike non-linear optimization algorithms derived by extending Newton's method, methods based on sequential linearization do not require the hessian matrix a hessian matrix can be very large in applications involving a large number of variables, therefore, constructing this matrix and its LU decomposition represents a large overhead. As a result, sequential approximation methods often perform better for applications where the number of variables are very large or the second derivatives of the function are very expensive to evaluate.

An important area of application of such sequential approximate techniques has been structural optimization, where function evaluation involves computationally expensive structural analysis. The evaluation of the gradient of the objective function and its Hessian matrix each represent significant computation. In addition, for combined shape and

topology optimization problems, the number of variables involved is very large and is dependent on the mesh density used for the finite element model. Since most mathematical programming methods do not perform well for such a large number of variables, optimality criteria methods have been used to solve the topology optimization problems by Suzuki and Kikuchi [Suzuki_91]. Among the mathematical programming algorithms, sequential approximation methods seem to be the most promising for such problems.

In Sequential Linear Programming, a non-linear programming problem is solved by solving a sequence of linear programs created by linear approximations of the objective and constraints. Schmit and Farshi [Schmit_74] suggested linearization in the reciprocal of the variables. Fluery and Briabant [Fluery_79, Fluery_86] proposed the use of mixed variables, that is to linearize with respect to some variables and with respect to the reciprocal of other variables in such a way as to construct a convex subproblem. This idea was further extended by Svanberg [Svanberg_87], where he suggests a method of moving asymptotes (MMA). This method can be interpreted as setting move limits on the variables using asymptotes. These asymptotes are moved to make the method stable and to expedite convergence. Svanberg suggests a set of heuristic rules for moving these asymptotes. Each sub-problem is solved by solving its dual problem that has a concave objective function and hence can be solved using conventional gradient methods such as the conjugate gradient method.

In this chapter, a sequential approximation method is proposed that is particularly suited for structural analysis problems with large number of variables. In this method, a sequence of sub-problems are generated by linearizing the objective function and setting move limits on the variables using logarithmic barrier functions. Therefore, we shall refer to this algorithm as moving barrier sequential linear programming (MBSLP). This method can be interpreted as sequential linear programming using the primal-dual technique [Monteiro_89a] to generate a descent direction and step size. Move limits on the variables are flexible and are moved each iteration in a fashion such that the upper and lower limits converge towards each other. The resulting algorithm is a general non-linear programming technique. The principal advantage of the method is that it does not require the exact solution of the sub-problems and hence requires very little computation per iteration. Numerical experiments suggests that the method exhibits fast convergence and requires fewer function and gradient evaluations than traditional Newton-like methods. It is insensitive to the scaling of the variables and performs well even for ill-conditioned problems if the hessian matrix is nearly diagonal.

The philosophy of sequential approximate optimization techniques and their evolution is summarized in section 4.2, with a brief description of some popular techniques. A sequential approximation method that uses logarithmic barriers to set move limits on the variables is introduced in section 4.3. The subproblem generated at each iteration for linear equality constrained problems is described along with the solution strategy and the criteria for moving the barriers. The algorithm requires a feasible starting point. A simple method for finding such a point using moving logarithmic barriers is described in section 4.4. The method is extended to handle nonlinear equality and inequality constrained optimization in section 4.5.

4.2. Sequential Optimization Algorithms

Sequential approximate optimization algorithms construct local approximations of the objective and constraint functions to create sub problems that are easier to solve. In structural optimization, the objective functions and/or constraints are not explicit functions of the design variables. Instead, the structural response must be determined by numerical methods such as the finite element method. A global approximation of the functions can be constructed by evaluating them at a large number of design points and fitting a polynomial approximation. Even though constructing such a global approximation would make the subsequent optimization fast, the construction of such a global approximation can be very expensive when the number of variables is large. Therefore, most sequential approximation algorithms construct local approximations using the objective and constraint function values and their gradients evaluated at the current best guess of the solution. Some of the common local approximations that have been used are linear approximations, reciprocal approximations and convex approximations constructed by using a hybrid of linear and reciprocal approximations.

Linear approximation is the simplest of the local approximation schemes and is based on the first order Taylor series expansion of the function. A nonlinear function f(**x**) is linearized about a point $\mathbf{x}_k$ as follows:

$$F_L(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^t(\mathbf{x} - \mathbf{x}_k) \tag{4.2.1}$$

Rather than linearize over the design variables it is possible to linearize with respect to intermediate variables. For example, if a set of intermediate variables are defined as functions of the design variables as:

$$\mathbf{x} = \mathbf{x}(\mathbf{y}), \quad \mathbf{y} \in R^n \tag{4.2.2}$$

Then a function f(**x**) can be linearized with respect to **y** as

$$F_I(\mathbf{y}) = f(\mathbf{x}(\mathbf{y}_k)) + \nabla_y f(\mathbf{x}(\mathbf{y}_k))^t(\mathbf{y} - \mathbf{y}_k) \tag{4.2.3}$$

where, ∇_y is the gradient with respect to the intermediate variables **y**.

In structural optimization, the reciprocal variables have been a very popular intermediate variable. The reason is that for many truss and frame sizing optimization problems, the stress and displacement functions are linear functions of the reciprocal of the design variables. As a result linearization with respect to reciprocal variables leads to accurate linearization. Reciprocal approximation can be obtained by linearizing with respect to the reciprocal intermediate variables.

Using $y_i = 1/x_i$, $i = 1,\ldots n$, in equation (4.2.3) and linearizing with respect to **y** yields the reciprocal approximation. This approximation can then be expressed in terms of **x** by substituting for **y** in terms of **x** to obtain the following,

$$F_R(\mathbf{x}) = f(\mathbf{x}_k) + \sum_{i=1}^{n}(x_i - x_{ik})\frac{x_{ik}}{x_i}\frac{\partial f}{\partial x_i}\bigg|_{x=x_k} \tag{4.2.4}$$

A hybrid of the linear and reciprocal approximations leads to the so call conservative approximation or mixed linearization, which can be expressed as,

$$F_C(\mathbf{x}) = f(\mathbf{x}_k) + \sum_{i=1}^{n} r_i(x_i - x_{ik})\frac{\partial f}{\partial x_i}\bigg|_{x=x_k} \tag{4.2.5}$$

$$\text{where, } r_i = \begin{cases} 1, & \text{if } x_i\dfrac{\partial f(\mathbf{x}_k)}{\partial x_i} \le 0 \\ \dfrac{x_{ik}}{x_i}, & \text{otherwise} \end{cases} \tag{4.2.6}$$

This approximation is referred to as the conservative approximation because when a constraint is expressed as $f(\mathbf{x})\ge 0$, then $F_C(\mathbf{x})$ is more conservative because $F_C(\mathbf{x}) \le F_L(\mathbf{x})$. The approximation $F_C(\mathbf{x})$ is constructed by setting $F_C(\mathbf{x})$ identical to $F_R(\mathbf{x})$ in the region where $F_L(\mathbf{x}) > F_R(\mathbf{x})$ and $F_C(\mathbf{x}) = F_L(\mathbf{x})$ otherwise. The approximate function constructed here is concave. One could convert it to a convex approximation by redefining r_i. This property makes conservative approximation popular, since the convexity / concavity of the functions can be used to design elegant and fast optimization algorithms.

Higher order approximations can also be constructed by expanding the function using Taylor series and including the higher order terms. Thus, one could create a quadratic approximation by including the second order terms of the Taylor series. Using the reciprocal of the variables in the quadratic approximation, one can construct the reciprocal quadratic approximation and so on.

Sequential approximate optimization algorithms construct a sequence of sub problems using approximations such as those cited above. The basic algorithm common to all sequential approximation algorithms may be summarized as:

1) Guess an initial value of the variables $\mathbf{x}_0$ at iteration k=0.

2) For each iteration k, evaluate the objective function f(**x**), the constraints **h**(**x**), **g**(**x**) and their gradients $\nabla f(\mathbf{x}), \nabla \mathbf{g}(\mathbf{x}), \nabla \mathbf{h}(\mathbf{x})$ at the design point $\mathbf{x}_k$.

3) Construct local approximations of the objective function and all the nonlinear constraints about the point $\mathbf{x}_k$ using the function values and gradients evaluated in step 2 and generate a subproblem defined in terms of these approximate functions. (Typically bounds are set on the variables in the sub problem's definition. These bounds or side constraints are referred to as move limits and they are updated each iteration.)

4) Solve the above subproblem to obtain its optimal point $\mathbf{x}_k^*$.

5) Test whether this point satisfies the convergence criteria for the original problem. Terminate the iterations if it does, otherwise use $\mathbf{x}_k^*$ as the starting point of the next iteration. That is, set $\mathbf{x}_{k+1} = \mathbf{x}_k^*$ and go to step 2.

Various sequential approximate optimization algorithms have been proposed. They differ from each other due to the differences in the local approximation schemes used and in the sub problem definition.

4.2.1. Sequential Linear Programming (SLP)

Sequential linear programming is the simplest and earliest sequential approximate optimization algorithm. It constructs sub problems using the linear approximation of the objective and constraints functions. Consider a nonlinear programming problem stated as

$$\begin{aligned} &\text{minimize } f_0(\mathbf{x}), \\ &\text{subject to } f_j(\mathbf{x}) \le 0,\ j=1,..r \end{aligned} \tag{4.2.7}$$

Sub problems are constructed for this problem as,

$$\min\ f_0(\mathbf{x}^k) + \sum_{i=1}^{n}(x_i - x_i^k)\left.\frac{\partial f_0}{\partial x_i}\right|_{x=x^k} \tag{4.2.8}$$

subject to,

$$f_j(\mathbf{x}^k) + \sum_{i=1}^{n}(x_i - x_i^k)\left.\frac{\partial f_j}{\partial x_i}\right|_{x=x^k} \le 0,\ j = 1,\ldots r \tag{4.2.9}$$

$$l_i^k \le x_i^k \le u_i^k,\ i = 1,\ldots n \tag{4.2.10}$$

The above problem is a standard linear programming (LP) problem that can be solved using standard LP packages that are very reliable and robust. The vectors l_i^k and u_i^k, i=1,..n are the move limits that set bounds on the variables. They are reset every iteration. One problem with SLP is that each iteration involves solving a LP completely and hence the computational cost per iteration is very high especially when there are a large number of variables. Moreover, one needs a good strategy to set move limits to ensure convergence. The move limits are tightened at each iteration so that the upper and lower limits move towards each other. Various different criteria have been proposed to determine when and by how much to shrink the move limits.

Schmit and Farshi [Schmit_74] proposed linearization in reciprocal variables to construct sub problems similar to those constructed above. These sub problems are simple nonlinear optimization problem that are typically inexpensive to solve relative to original nonlinear problem. In the design of statically determinate trusses and frames, the stresses and displacements are linear functions of the reciprocal of the design variables, namely the cross-sectional sizing variables. This was the original motivation for linearizing with respect to the reciprocal variables, however, this technique was found to be beneficial for other structural optimization problems as well.

4.2.2. Convex Linearization (CONLIN)

Convex linearization methods construct a sequence of convex sub-problems whose solutions converge to the solution of the original nonlinear programming problem. The sub-problem is constructed by mixed linearization that was described earlier. For the nonlinear program (4.2.7) described above, the objective and constraint functions are locally approximated using conservative approximation such that all the functions in the subproblem are convex. The subproblem may be stated as :

$$\min \sum_{i=1}^{n} \delta_i^+ f_{i0} x_i - \sum_{i=1}^{n} \delta_i^- \frac{f_{i0}}{x_i} - d_o \tag{4.2.11}$$

subject to,

$$\sum_{i=1}^{n} \delta_i^+ f_{ij} x_i - \sum_{i=1}^{n} \delta_i^- \frac{f_{ij}}{x_i} \le d_j,\ j = 1,\ldots r \tag{4.2.12}$$

where,

$$\delta_i^+ = 1,\ \delta_i^- = 0,\ f_{ij} = \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k}, \quad \text{if } \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k} > 0 \tag{4.2.13}$$

$$\delta_i^+ = 0,\ \delta_i^- = 1,\ f_{ij} = (x_{ik})^2 \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k}, \quad \text{if } \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k} < 0 \tag{4.2.14}$$

$$d_j = \sum_{i=1}^{n} x_{ik} \left|\left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k}\right| - f_j(x_k), \quad j = 0,\ldots r \tag{4.2.15}$$

The inequalities in the above subproblem are conservative approximations to the inequalities $f_j(\mathbf{x}) \le 0$, j=1,..r. In addition, the objective and constraint functions of this subproblem are convex [Fluery_86]. Therefore, the method has been referred to as convex linearization (or CONLIN) method. Taking advantage of the convexity and seperability of the subproblem, various dual methods have been proposed [Fluery_79, Fluery_86] to solve these sub problems efficiently.

4.2.3. Method of Moving Asymptotes (MMA)

The method of moving asymptotes is a generalization of the convex linearization method that provides flexibility in controlling the degree of convexity and conservativeness of the approximation. The local approximation used in this method linearizes the functions with respect to the intermediate variables $1/(U_i - x_i)$ or $1/(x_i - L_i)$ depending on the sign of the partial derivatives of the function with respect to x_i. For the nonlinear problem (NLP1), the subproblem for the kth iteration may be stated as:

$$\min \sum_{i=1}^{n} \delta_i^+ \frac{f_{i0}}{U_i - x_i} - \sum_{i=1}^{n} \delta_i^- \frac{f_{i0}}{x_i - L_i} - d_o \tag{4.2.16}$$

subject to,

$$\sum_{i=1}^{n} \delta_i^+ \frac{f_{ij}}{U_i - x_i} - \sum_{i=1}^{n} \delta_i^- \frac{f_{ij}}{x_i - L_i} \le d_j,\ j = 1,\ldots r \tag{4.2.17}$$

where,

$$\delta_i^+ = 1,\ \delta_i^- = 0,\ f_{ij} = (U_i - x_{ik})^2 \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k},\quad \text{if } \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k} > 0 \tag{4.2.18}$$

$$\delta_i^+ = 0,\ \delta_i^- = 1,\ f_{ij} = (x_{ik} - L_i)^2 \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k},\quad \text{if } \left.\frac{\partial f_j}{\partial x_i}\right|_{x=x_k} < 0 \tag{4.2.19}$$

$$d_j = \sum_{i=1}^{n} \delta_i^+ (U_i - x_i) \left.\frac{\partial f_j}{\partial x_i}\right|_{x_k} - \sum_{i=1}^{n} \delta_i^- (x_{ik} - L_i) \left.\frac{\partial f_j}{\partial x_i}\right|_{x_k} - f_j(\mathbf{x}_k) \tag{4.2.20}$$

The constants L_i and U_i are reset at each iteration and they define the asymptotes of the approximate functions used in the subproblem. Therefore, this method can be interpreted as convex linearization with move limits set by these asymptotes for each subproblem. Also we note that MMA reduces to convex linearization method if we set L_i=0 and U_i=+∞. When the L_i and U_i are set closer to each other the degree of convexity and conservativeness of the approximation increases.

4.3. Moving Barrier Sequential Linear Programming (MBSLP)

In this section, we propose a sequential optimization algorithm that we refer to as the moving barrier sequential linear programming. In this algorithm, the functions are approximated as a sum of the linear approximation of the function and logarithmic barrier terms that tend to infinity near the move limits on the variables. This approximation is convex and we refer to it as the logarithmic barrier approximation. The sub-problems generated using this approximation for linearly constrained problems and solution methods for these sub problems are described in this section.

4.3.1. Subproblem definition

Consider a optimization problem with linear constraints and side constraints as described below,

$$(\mathbf{P}) : \text{Min}\quad f(\mathbf{x}),\ \mathbf{x} \in R^n, f: R^n \to R \tag{4.3.1}$$

subject to,

$$\begin{aligned} &\mathbf{Ax} = \mathbf{b},\ \mathbf{A} \in R^{m \times n},\ \mathbf{b} \in R^m \\ &l_i \le x_i \le u_i,\ i = 1,\ldots..,n \end{aligned} \tag{4.3.2}$$

The objective function f is a nonlinear function. The vectors **l** and **u** set the side constraints on the variables, written succinctly as, $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$. In the solution of the above

optimization problem the side constraints are treated differently than the other linear constraints. A sequence of sub-problems are generated of the form,

$$(\mathbf{P_k}) : \text{Min } F_k(\mathbf{x}) \text{ , such that, } \quad \mathbf{x} \in S_k \tag{4.3.3}$$

$$S_k = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b} \; ; \; \hat{\mathbf{l}}^k < \mathbf{x} < \hat{\mathbf{u}}^k\} \tag{4.3.4}$$

$$F_k(\mathbf{x}) = \nabla f(\mathbf{x}^k)^t \mathbf{x} - \varepsilon_k \sum_{i=1}^{n} \ln(x_i - \hat{l}_i^k) - \varepsilon_k \sum_{i=1}^{n} \ln(\hat{u}_i^k - x_i) \tag{4.3.5}$$

The function $F_k(\mathbf{x})$ is the objective function of the sub problem $(\mathbf{P_k})$ generated at the k^{th} iteration. This function is constructed by logarithmic barrier approximation of the original objective function $f(\mathbf{x})$. This approximate function is strictly convex since it is the sum of terms linear and logarithmic in the design variables. Assuming that S_k is a non-empty and bounded set, there exists a unique global minimum for the above sub-problem. Note that the side constraints for the sub-problem $(\mathbf{P_k})$ are not the same as that for the original problem $(\mathbf{P})$. Instead, $\hat{\mathbf{l}}^k$ and $\hat{\mathbf{u}}^k$ are flexible move limits that satisfy the condition,

$$\mathbf{l} \le \hat{\mathbf{l}}^k < \mathbf{x} < \hat{\mathbf{u}}^k \le \mathbf{u} \tag{4.3.6}$$

These move limits serve to limit the step size taken at each iteration k. A criterion for selecting their value is given later. This criterion causes the upper and lower move limits to converge towards each other and the optimal solution. If the gradient vector $\mathbf{c} = \nabla f(\mathbf{x}^k)$ is taken to be a constant, the above sub-problem can be viewed as linear programming, using barrier methods to enforce the move limits. The logarithmic barrier function has been found to perform well for the primal-dual method (Monteiro_89a) of linear programming. The primal dual method is a interior point method for linear programming that uses logarithmic barriers to impose side constraints. In one dimension the objective function of the subproblem may be represented as shown in figure 4.1. The gradient of the function is evaluated at the current value of the variables $\mathbf{x}^k$. The straight line tangent to the function $f(\mathbf{x})$ at the this point is the linear approximation of the function. The objective function of the sub-problem, $F_k(\mathbf{x})$, is nearly equal to the linear approximation in the neighborhood of the point $\mathbf{x}^k$, but due to the logarithmic terms its values rises sharply near the move limits $\mathbf{x} = \hat{\mathbf{l}}^k$ and $\mathbf{x} = \hat{\mathbf{u}}^k$. Therefore, these logarithmic terms behave as barriers that prevent the variables from violating the move limits during the optimization of the sub problem.

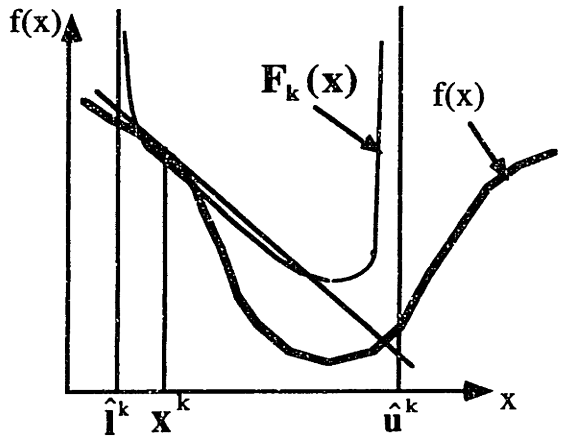


Figure 4.1. Objective function of the subproblem for one-dimensional case

4.3.2. The dual of the subproblem

In the sub problem defined in the previous section, if we neglect the logarithmic barrier terms and treat the gradient vector **c** as a constant, then the sub problem becomes a linear program. The dual problem for this linear program, $\min \mathbf{c}^t\mathbf{x}$, $\mathbf{x} \in S_k$ is defined in this section. The dual variables are the Lagrange multipliers of the linear program. The variables defined in connection with this dual problem are useful in describing the solution scheme in the next section. The dual problem of the linear program can be defined as

$$(\mathbf{D_k}) : \max\ q(\lambda) \tag{4.3.7}$$

$$\begin{aligned} q(\lambda) &= \underset{l \le x \le u}{\text{infimum}} \{\mathbf{c}^t\mathbf{x} + \lambda^t(\mathbf{A}\mathbf{x} - \mathbf{b})\} \\ &= (\mathbf{c}^t + \lambda^t\mathbf{A})\hat{\mathbf{x}} - \lambda^t\mathbf{b} \\ &= \mathbf{z}^t\hat{\mathbf{x}} - \lambda^t\mathbf{b} \end{aligned} \tag{4.3.8}$$

where,

$$\mathbf{z} = (\mathbf{c} + \mathbf{A}^t\lambda) \in R^n \text{ and } \hat{x}_i = \begin{Bmatrix} l_i & \text{if } z_i > 0 \\ u_i & \text{otherwise} \end{Bmatrix} \tag{4.3.9}$$

λ is the vector of Lagrange multipliers that serve as the variables of the dual problem. The vector **z** is a set of intermediate variables that are linearly dependent on the dual variables λ.

4.3.3. Solution strategy

The optimality criteria for the sub-problem $(\mathbf{P_k})$ can be derived as,

$$
\begin{aligned}
&(x_i - \hat{l}_i^k)(\hat{u}_i^k - x_i)z_i - \varepsilon_k(\hat{u}_i^k + \hat{l}_i^k - 2x_i) = 0,\ i = 1....n \\
&\mathbf{Ax} = \mathbf{b} \\
&\mathbf{z} - \mathbf{A}^t\lambda = \mathbf{c},
\end{aligned}
\tag{4.3.10}
$$

where,

$$
\mathbf{z} \in R^n,\quad \mathbf{c} = \nabla f(\mathbf{x}^k) \in R^n \quad \text{and} \quad \lambda \in R^m \tag{4.3.11}
$$

The optimality criteria are a set of nonlinear simultaneous equations. Due to the convexity of the sub-problem, we know that the solution of these equations would yield the unique global minimum of the sub-problem. These equations could be solved using Newton's method. In doing so we treat $\mathbf{c} = \nabla f(\mathbf{x}^k)$ as a constant vector, thereby making a linear approximation of the objective function. Using the standard first order Taylor series expansion, we get the following linear simultaneous equations for evaluating the Newton descent direction.

$$
\begin{aligned}
&(x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)\Delta z_i + \left|(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)z_i^k\right|\Delta x_i + 2\varepsilon_k \Delta x_i = -v_i^k, \quad i = 1....n \\
&\mathbf{A}\Delta\mathbf{x} = 0 \\
&\Delta\mathbf{z} - \mathbf{A}^t\Delta\lambda = 0
\end{aligned}
\tag{4.3.12}
$$

where,

$$
v_i^k = (x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)z_i^k - \varepsilon_k(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k) \tag{4.3.13}
$$

The above equations can be solved easily using a linear equation solver. However, we note that the first 'n' equations of (4.3.12) are decoupled. We make use of this special structure of the equations to calculate the solution efficiently. The following notation simplifies the algebraic manipulation of the above equations. Let $\mathbf{X} \in R^{n\times n}$ and $\mathbf{Z} \in R^{n\times n}$ be diagonal matrices defined as:

$$
X_{ii} = (x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k),\quad X_{ij} = 0, i \neq j \tag{4.3.14}
$$

$$
Z_{ii} = \left|(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)z_i^k\right| + 2\varepsilon_k,\quad Z_{ij} = 0, i \neq j \tag{4.3.15}
$$

Equations (4.3.12) and (4.3.13) can then be rewritten as;

$$
\mathbf{X}\Delta\mathbf{z} + \mathbf{Z}\Delta\mathbf{x} = -\mathbf{v},\ \text{where, } \mathbf{v} = \mathbf{X}\mathbf{z}^k - \varepsilon_k(\hat{\mathbf{u}}^k + \hat{\mathbf{l}}^k - 2\mathbf{x}^k) \tag{4.3.16}
$$

$$
\mathbf{A}\Delta\mathbf{x} = \mathbf{0} \tag{4.3.17}
$$

$$
\Delta\mathbf{z} - \mathbf{A}^t\Delta\lambda = 0 \tag{4.3.18}
$$

Multiplying equation (4.3.16) by $\mathbf{AZ}^{-1}$ and substituting equations (4.3.17) & (4.3.18) into (4.3.16), we get,

$$\begin{aligned}
&\Delta\lambda = -(\mathbf{ADA}^t)^{-1}\mathbf{AV} \\
&\Delta\mathbf{z} = \mathbf{A}^t\Delta\lambda \\
&\Delta\mathbf{x} = -\mathbf{V} - \mathbf{D}\Delta\mathbf{z}, \quad \mathbf{V} \in R^n \text{ and } \mathbf{D} \in R^{n\times n}
\end{aligned} \tag{4.3.19}$$

where, $\mathbf{D} = \mathbf{Z}^{-1}\mathbf{X}$ and $\mathbf{V} = \mathbf{Z}^{-1}\mathbf{v}$ which can be written out as,

$$\begin{aligned}
D_{ii} &= \frac{(x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)}{\left|(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)z_i^k\right| + 2\varepsilon_k}, \quad D_{ij} = 0, \ i \neq j, \quad i,j = 1....n \\
V_i &= \frac{v_i}{\left|(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)z_i^k\right| + 2\varepsilon_k}, \ i = 1....n
\end{aligned} \tag{4.3.20}$$

The variables are updated as follows,

$$\begin{aligned}
\mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k\Delta x \\
\mathbf{z}^{k+1} &= \mathbf{z}^k + \alpha_k\Delta\mathbf{z} \\
\lambda^{k+1} &= \lambda^k + \alpha_k\Delta\lambda
\end{aligned} \tag{4.3.21}$$

The step size α_k is selected such that the new values of the variables are feasible, i.e., $\hat{\mathbf{l}}^k < \mathbf{x} < \hat{\mathbf{u}}^k$. The step size may therefore be computed as follows,

$$\alpha_k = \min_{i=1....n}\{\tilde{\alpha}_i\} \tag{4.3.22}$$

where,

$$\tilde{\alpha}_i = \left\{ \begin{array}{ll} \dfrac{\hat{l}_i^k - x_i^k}{\Delta x_i^k} & \text{if } \Delta x_i < 0 \\ \dfrac{\hat{u}_i^k - x_i^k}{\Delta x_i^k} & \text{otherwise} \end{array} \right\}, \ i=1...n \tag{4.3.23}$$

The Newton iterations derived above may be used to solve the optimality criteria equations and thereby obtain the global minimum of the sub-problem ($\mathbf{P_k}$). However, since the minimum of the subproblem is not the same as the minimum of our original problem, we do not solve the subproblem exactly. Instead, we restrict the number of Newton iterations to the minimum required to generate a descent direction. For unconstrained optimization such a direction is found at the very first iteration. However, for constrained

optimization, if our initial guesses for the Lagrange multipliers are not close to the real values, more than one iteration is required. If this is the case we use the above iterations to update λ and **z**, but leave **x** unchanged. Once a descent direction is found, the variables are updated and the subproblem is redefined by re-evaluating the functions and gradients and by resetting the move limits. Typically very few Newton steps are required per sub-problem. As a result the computation per iteration is reduced as compared to other sequential programming algorithms where the sub problems have to be completely solved using dual methods (see [Svanberg_87]). Note that at each iteration we need to solve only 'm' simultaneous equations to obtain $\Delta\lambda$ in equation (4.3.19). Also note that only one function and gradient evaluation is required per iteration to define the sub-problem. However, once the descent direction and step size are found it is beneficial to use Armijo's rule to reduce the step size further if necessary. When Armijo's rule is used to reduce step size, one additional function evaluation is required per step size reduction. The move limits set by the barrier function usually makes further step size reduction unnecessary in most iterations so that Armijo's rule serves merely as a safety check. The Armijo rule used in our implementation is given below. We set the step size $s_k = \beta^{m_k}\alpha_k$, where we chose a fixed scalar β such that $0 < \beta < 1$ and m_k is the smallest positive integer for which the following relation holds.

$$f(x_k) - f(x_k + \beta^{m_k}\alpha_k \Delta x_k) \geq -\sigma\beta^{m_k}\alpha_k \nabla f(x_k)^t \Delta x_k \qquad (4.3.24)$$

This algorithm can be extended to handle inequality constraints of the form $\mathbf{Cx} \leq \mathbf{d}$, , $\mathbf{C} \in R^{rxn}, \mathbf{d} \in R^r$ by using slack variables to convert them to equality constraints of the form $\mathbf{Cx} + s = \mathbf{d}$, $s \in R^r$, $s_i \geq 0$. In the next section, we give the details of handling inequality constraints for very general nonlinear programs.

4.3.4. Criteria for moving the barriers

The move limits are set at each sub-problem in such a way that the variables stay within the feasible domain of the original optimization problem (**P**). The upper and lower limits are moved such that they converge towards each other and the feasible region for the sub-problems shrinks progressively. The move limits help to stabilize the algorithm and prevent the step size from being excessive. The values of the move limits are initialized to be the same as the side constraints on the original optimization problem (**P**), that is, $\hat{l}_i^0 = l_i$ and $\hat{u}_i^0 = u_i$.

Many update schemes are possible for resetting the move limits at each iteration. Here we give a scheme that was found to perform very well for linearly constrained problems

where a large number of the side constraints are likely to become active. In this scheme, the move limits are updated as follows,

if $\Delta x_i > 0$,

$$\hat{l}_i^{k+1} = x_i^k \quad \text{(Update lower bounds)} \tag{4.3.25}$$

else if $\Delta x_i < 0$,

$$u_i^{k+1} = x_i^k \quad \text{(Update upper bounds)} \tag{4.3.26}$$

This criterion is based on the idea that if the value $\Delta x_i > 0$, then x_i is increasing and by updating the lower limit as above, we prevent the x_i from subsequently reducing below x_i^k. This stabilizes the iterations and if the value of the variable is oscillating about its optimal value, this ensures that the step size is progressively reduced until both the upper and lower move limits converge towards the optimal value. Figure 4.2 illustrates this move criterion for a unconstrained two-dimensional optimization. The contours of the function are shown in the design space. In this case, the descent direction is such that both the variables are increasing and therefore the lower move limits are updated as shown and the upper bounds are unaffected.

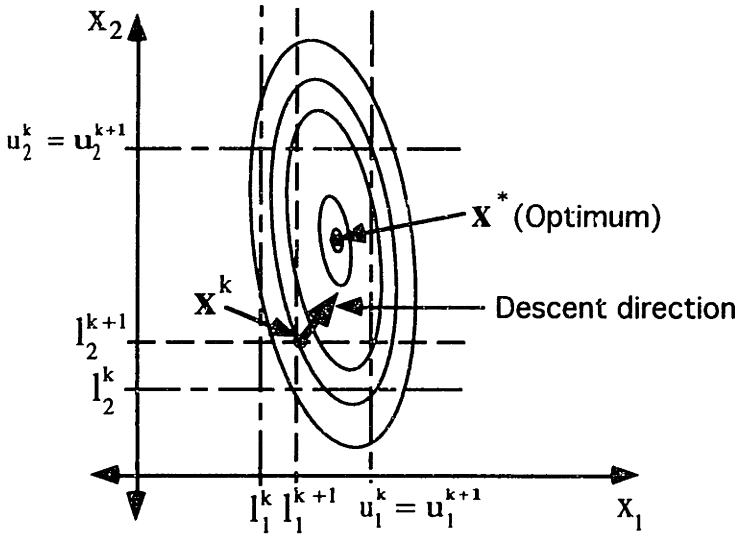


Figure 4.2. Resetting the move limits

This criterion ensures convergence if the objective function is strictly convex. However, if the function is not convex, the criterion (4.3.25, 4.3.26) alone does not work. For example, when the contours of the objective function are non-convex, it is possible for

the Δx_i to be positive but the optimal value of the x_i to be less than x_i^k and vice versa. This is illustrated in figure 4.3.

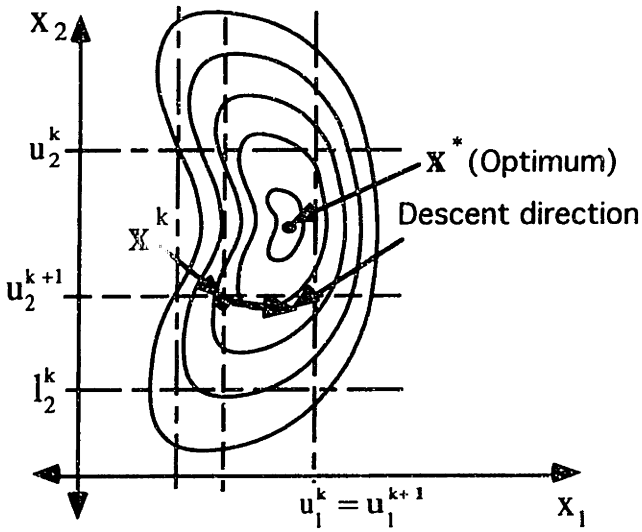


Figure 4.3. Failure of resetting criterion for nonconvex functions

The contours of the function are non convex in the example shown in figure 4.3. In this case, the descent direction is such that the variable x_2 decreases causing the upper move limit to be reset as shown. The optimum point $\mathbf{x}^*$ now lies outside move limits. In subsequent iterations, as the variable $\mathbf{x}^k$ tries to move towards the optimum, the upper move limit will become active. Since the move limits are artificial bounds set on the variables, they should not become active constraints at the optimum. Therefore, if a move limit becomes active, it is an indication that a situation such as the one illustrated in figure 4.3 has occurred. Therefore, we need a criterion to detect when a moving limit becomes active and if it does, the move limit is pushed back. The criterion may be stated as follows,

if $(z_i > 0)$ and $(v_i < 0)$,

$$\hat{l}_i^{k+1} = \max\left\{\left(\hat{l}_i^k - (\hat{u}_i^k - \hat{l}_i^k)\right),\ l_i\right\} \tag{4.3.27}$$

else if $(z_i < 0)$ and $(v_i > 0)$,

$$\hat{u}_i^{k+1} = \min\left\{\left(\hat{u}_i^k + (\hat{u}_i^k - \hat{l}_i^k)\right),\ u_i\right\} \tag{4.3.28}$$

This criterion is based on the observation that when $z_i > 0$, the variables x_i should decrease and tend towards its lower limit. For the unconstrained case, $\Delta x_i = -v_i/Z_{ii}$, therefore, v_i should be greater than zero in order for Δx_i to decrease ($Z_{ii}>0$). In equation (4.3.13), we defined v_i^k as:

$$v_i^k = (x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)z_i^k - \varepsilon_k(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)$$

Clearly, the sign of v_i^k would be identical to z_i^k, when the first term in the right hand side is dominant. The factor ε_k tends to zero as the iterations progress and the second term on the r.h.s of equation (4.3.13) also tends toward zero. However, if a bound is active or nearly active then the first term vanishes and the sign of v_i^k is determined by the second term. In this case, we push back the lower move limit using the update formula in equation (4.3.27) and (4.3.28).

The value of the constant ε_k is set each iteration such that it tends to zero. $\Delta\mathbf{x}$ is a descent direction only if $\mathbf{c}^t\Delta\mathbf{x} < 0$. Using the expression for $\Delta\mathbf{x}$ in equation (4.3.19), we get,

$$\mathbf{c}^t\Delta\mathbf{x} = \left(\varepsilon_k\sum_{i=1}^{n} c_i(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k) - \sum_{i=1}^{n} c_i(z_i^k + \Delta z_i)X_{ii}\right)\Big/ Z_{ii} \qquad (4.3.29)$$

Clearly, the value of ε_k influences $\Delta\mathbf{x}$ so that for large values of ε_k, $\Delta\mathbf{x}$ may not be a descent direction. In our implementation, we set ε_k at each iteration such that,

$$\varepsilon_k = \frac{g_k}{2n}, \text{ where } g_k = \mathbf{z}^t(\mathbf{x} - \hat{\mathbf{x}}) \qquad (4.3.30)$$

$\mathbf{z}$ and $\hat{\mathbf{x}}$ are defined in equation (4.3.9). Note that g_k is the duality gap between the linear program $\min \mathbf{c}^t\mathbf{x}$, $\mathbf{x} \in S_k$ and its dual $\max q(\lambda)$ defined in equation (4.3.8). The duality gap decreases when $\mathbf{z}$ becomes very small and when the upper and lower move limits of the variables approach each other. Therefore, resetting ε_k by equation (4.3.30) ensures that its value decreases each iteration as the move limits converge towards each other. The value of ε_k obtained by equation (4.3.30) does not guarantee that $\Delta\mathbf{x}$ is a descent direction. When some variables are very close to the move limits such that $X_{ii} = (x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)$ tends to zero then, it is possible that the terms in the first summation of equation (4.3.29) dominate causing $\Delta\mathbf{x}$ to be a non-descent direction. However, as noted earlier, the criteria for updating the move limits (4.3.27) and (4.3.28) prevents this by moving back the move limits whenever they tend to be active constraints. Therefore, the criteria for resetting ε_k and the move limits together ensure that a descent direction is obtained at each iteration.

4.4. Finding an initial feasible point

The algorithm described above is an interior point method since we assume that the current guess of the solution, $\mathbf{x}^k$, is always within in the feasible region. The initial guess of the solution must therefore be a feasible point. During the subsequent iterations, we get a sequence of points that are within the feasible region due to the move limits set by the barrier function. An initial feasible point can be found using a simple modification to the moving barrier technique.

An arbitrary starting point $\mathbf{x}^0$ can be projected on the hyper plane $\mathbf{Ax} = \mathbf{b}$ by minimizing $\|\mathbf{x} - \mathbf{x}^0\|^2$ subject to $\mathbf{Ax} = \mathbf{b}$. The projected point $\tilde{\mathbf{x}}^0$ is obtained as

$$\tilde{\mathbf{x}}^0 = \mathbf{x}^0 - \mathbf{A}^t(\mathbf{A}\mathbf{A}^t)^{-1}(\mathbf{A}\mathbf{x}^0 - \mathbf{b}) \tag{4.4.1}$$

The analytical center of the polyhedra $S = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b} \; ; \; \hat{\mathbf{l}}^k < \mathbf{x} < \hat{\mathbf{u}}^k\}$ is defined as the unique minimum over S of the convex function F_b defined below.

$$\mathbf{P}_b\text{: Min } F_b(\mathbf{x}) = -\varepsilon_k\left(\sum_{i=1}^{n}\ln(x_i - \hat{l}_i^{\,k}) + \sum_{i=1}^{n}\ln(\hat{u}_i^{\,k} - x_i)\right), \; x \in S \tag{4.4.2}$$

The analytical center is an interior point of the polyhedra S. If the point $\tilde{\mathbf{x}}^0$ does not satisfy the feasibility requirement $\mathbf{l} < \tilde{\mathbf{x}}^0 < \mathbf{u}$ of the original problem (**P**), then move limits of the problem ($\mathbf{P}_b$) are set such that $\hat{\mathbf{l}} < \tilde{\mathbf{x}}^0 < \hat{\mathbf{u}}$ using equations (4.4.3). The method used in section 4.2, to find a descent direction for the sub problems ($\mathbf{P}_k$) can also be used for the optimization problem ($\mathbf{P}_b$). Indeed after setting $\mathbf{c} = \mathbf{0}$ equations (4.3.14), (4.3.15) and (4.3.16) yields a descent direction for ($\mathbf{P}_b$). In practice large values for ε_k leads to faster convergence towards the analytical center. After updating the value of $\tilde{\mathbf{x}}^0$ using the descent direction, the move limits may be reset as,

$$\begin{aligned}
&\hat{l}_i^k = \tilde{x}_i^k - \delta \; ; \; \hat{u}_i^k = u_i \text{ if } x_i \le l_i \\
&\hat{u}_i^k = \tilde{x}_i^k + \delta \; ; \; \hat{l}_i^k = l_i \text{ if } x_i \ge u_i \\
&\hat{u}_i^k = u_i \; ; \; \hat{l}_i^k = l_i \; ; \text{ if } l_i \le x_i \le u_i, \; i = 1...n
\end{aligned} \tag{4.4.3}$$

where δ is a small positive real number. The concept is illustrated for the one-dimensional case in figure 4.4. The current guess for the variable, x^k, does not lie within the side constraints $l < x < u$. The move limits are set such that it encloses x^k and the lower limit $\hat{l}^k$ is nearly equal to x^k. When $F_b(x)$ is minimized, each iteration carries the variable closer to the analytical center of the set $S^k = \{x \mid \hat{l}^k < x < \hat{u}^k\}$. The move limit is reset after each iteration using equation (4.4.3) so that the move limit $\hat{l}^k$ moves closer to the new value of the variable x^k. The iterations are continued until x^k is in the feasible region $l<x<u$.

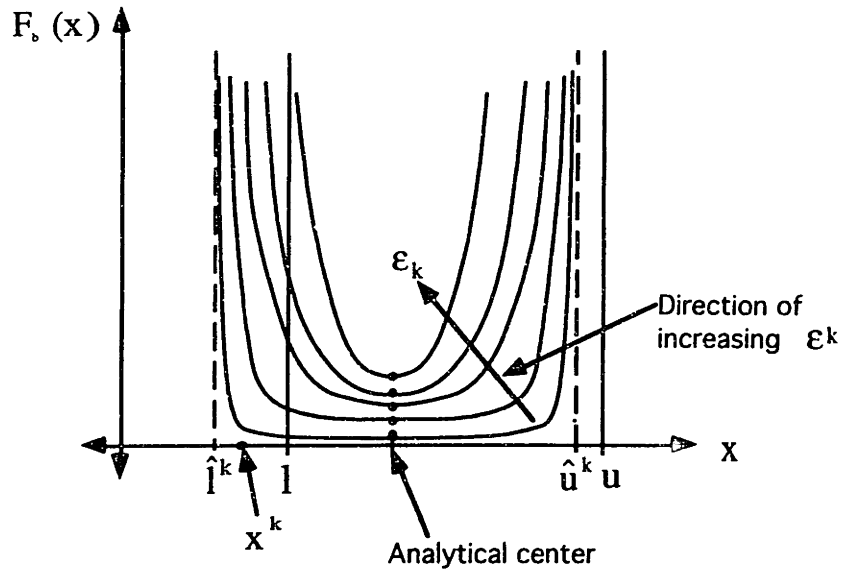


Figure 4.4. Finding a feasible point (1D case)

For linearly constrained problems, the value of $\tilde{\mathbf{x}}^0$ is changed at each iteration such that it moves away from the move limits along the hyper plane $\mathbf{Ax} = \mathbf{b}$. By resetting the move limits using equations (4.4.3), the move limits are again moved closer to the variable $\tilde{\mathbf{x}}^0$. As the iterations are continued the move limits approach the actual bounds on the variables $\mathbf{l}$ and $\mathbf{u}$. The iterations are stopped when the feasibility conditions $\mathbf{l} < \tilde{\mathbf{x}}^k < \mathbf{u}$ are satisfied.

Figure 4.2. illustrates a three-dimensional example where the linear constraint is a plane and the feasible region is a triangle on this plane whose edges are defined by the lower bounds on the variables. An initial guess is projected on to this plane. If the projected point does not lie inside the feasible triangle, the lower move limits for the variable are adjusted so that the projected point now lies inside the feasible region defined by the new move limits. Subsequent iterations towards the center of the analytical center of the feasible triangle gives us a sequence of points lying on the plane and tending towards the analytical center. The iterations are stopped when we obtain a point that lies inside the feasible triangle.

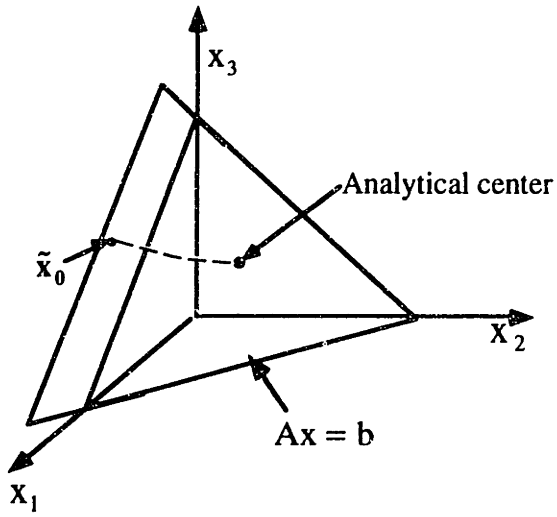


Figure 4.5. Finding a feasible point (3D illustrative example)

4.5. Extending MBSLP to handle nonlinear constraints

In the last two sections, we introduced a sequential approximate algorithm for optimizing linearly constrained optimization problems. In this section, we extend the method to solve general nonlinear programming problems with nonlinear equality as well as inequality constraints. A general nonlinear programming problem can be stated as,

$$(\text{NLP}): \text{Min } f(\mathbf{x}),\ \mathbf{x} \in R^n,\ f: R^n \to R \tag{4.5.1}$$

subject to,

$$\mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{h}: R^n \to R^m$$

$$\mathbf{g}(\mathbf{x}) \le \mathbf{0},\ \mathbf{g}: R^n \to R^r \tag{4.5.2}$$

$$l_i \le x_i \le u_i,\ i = 1,\ldots,n$$

where $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are nonlinear functions that express the equality and inequality constraints respectively. As before, $f(\mathbf{x})$ is the objective function which may be a nonlinear function of the variables. The vectors $\mathbf{l}$ and $\mathbf{u}$ are used to describe the side constraints on the variables. An approximation to the above nonlinear program may be constructed as follows,

$$\text{Min } \nabla f(\mathbf{x}_k)^t \mathbf{x} - \varepsilon_k \sum_{i=1}^{n} \ln(x_i - l_i) - \varepsilon_k \sum_{i=1}^{n} \ln(u_i - x_i) \tag{4.5.3}$$

subject to,

$$\mathbf{h}(\mathbf{x}_k)+\nabla\mathbf{h}(\mathbf{x}_k)^t(\mathbf{x}-\mathbf{x}_k)=\mathbf{0}$$
$$\mathbf{g}(\mathbf{x}_k)+\nabla\mathbf{g}(\mathbf{x}_k)^t(\mathbf{x}-\mathbf{x}_k)\le\mathbf{0} \qquad (4.5.4)$$
$$l_i \le x_i \le u_i,\ i=1,\ldots,n$$

To apply a method similar to the one developed in section 3, we convert inequality constraints into equalities using slack variables **s**. The sub problems may then be defined as:

$$P_k : \text{Min } F_k(\mathbf{x}_k) \qquad (4.5.5)$$

subject to,

$$\mathbf{A}\mathbf{x}=\mathbf{b} \qquad (4.5.6)$$
$$\mathbf{C}\mathbf{x}+\mathbf{s}=\mathbf{d},\ \mathbf{s}\in R^r \qquad (4.5.7)$$
$$l_i^k \le x_i \le u_i^k,\ i=1,\ldots n \qquad (4.5.8)$$
$$s_j \ge 0,\ j=1,..r \qquad (4.5.9)$$

where,

$$F_k(\mathbf{x}_k)=\mathbf{c}^t\mathbf{x}-\varepsilon_{1k}\sum_{i=1}^{n}\ln(x_i-\hat{l}_i^k)-\varepsilon_{1k}\sum_{i=1}^{n}\ln(\hat{u}_i^k-x_i)-\varepsilon_{2k}\sum_{i=1}^{r}\ln(s_i) \qquad (4.5.10)$$
$$\mathbf{A}=\nabla\mathbf{h}(\mathbf{x}_k)^t,\ \mathbf{A}\in R^{m\times n} \qquad (4.5.11)$$
$$\mathbf{C}=\nabla\mathbf{g}(\mathbf{x}_k)^t,\ \mathbf{C}\in R^{r\times n} \qquad (4.5.12)$$
$$\mathbf{b}=\nabla\mathbf{h}(\mathbf{x}_k)^t\mathbf{x}_k-\mathbf{h}(\mathbf{x}_k),\ \mathbf{b}\in R^m \qquad (4.5.13)$$
$$\mathbf{d}=\nabla\mathbf{g}(\mathbf{x}_k)^t\mathbf{x}_k-\mathbf{g}(\mathbf{x}_k),\ \mathbf{d}\in R^r \qquad (4.5.14)$$
$$\mathbf{c}=\nabla f(\mathbf{x}_k) \qquad (4.5.15)$$

By linearizing the nonlinear constraint equations about the current value of the variables, we now have a linearly constrained subproblem. The objective function of the subproblem $F_k(\mathbf{x}_k)$ is strictly convex and therefore we have a convex subproblem that has a unique minimum. Inequality constraints have been converted to equality constraints using the slack variables **s**. Note that logarithmic terms corresponding to the slack variables can be thought of as barriers that prevent the violation of the linearized inequality constraints. As before $\mathbf{l}^k$ and $\mathbf{u}^k$ are the move limits on the variables. The criteria for setting their value at each iteration is described later. Techniques similar to those developed in section 4.2 can be used to solve this subproblem. Using the Kuhn-Tucker conditions (see Appendix A) the optimality criteria for the subproblem may be stated as:

$$c_i - \varepsilon_{1k}\frac{(\hat{u}_i^k + \hat{l}_i^k - 2x_i)}{(x_i - \hat{l}_i^k)(\hat{u}_i^k - x_i)} + \sum_{j=1}^{m} A_{ji}\lambda_j + \sum_{j=1}^{r} C_{ji}\mu_j = 0, \quad i = 1,\ldots n \tag{4.5.16}$$

$$-\frac{\varepsilon_{2k}}{s_i} + \mu_i = 0, \quad i = 1,\ldots r \tag{4.5.17}$$

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{Cx} + \mathbf{s} = \mathbf{d} \tag{4.5.18}$$

$\lambda \in R^m$ and $\mu \in R^r$ are the Lagrange multipliers. The above nonlinear equations can be solved for the variables $\mathbf{x}$, λ, μ and $\mathbf{s}$ using Newton's method. However, as before we are not interested in the exact solution of the sub-problem and therefore carry out only the minimum number of iterations required to find a descent direction. For notational convenience we define, $\mathbf{z} \in R^n$ as $\mathbf{z} = \mathbf{c} + \mathbf{A}^t\lambda + \mathbf{C}^t\mu$. The variables may be updated using Newton iterations and the descent direction and the step size may be calculated as follows:

$$(x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)\Delta z_i + \{\left|(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)z_i^k\right| + 2\varepsilon_k\}\Delta x_i = -v_i^k, \quad i = 1,..n \tag{4.5.19}$$

$$s_i^k\Delta\mu_i + \mu_i\Delta s_i = -s_i^k\mu_i + \varepsilon_{2k} = -w_i^k \tag{4.5.20}$$

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{0} \tag{4.5.21}$$

$$\mathbf{C}\Delta\mathbf{x} + \Delta\mathbf{s} = \mathbf{0} \tag{4.5.22}$$

$$\Delta\mathbf{z} - \mathbf{A}^t\Delta\lambda - \mathbf{C}^t\Delta\mu = \mathbf{0} \tag{4.5.23}$$

where, $v_i^k = (x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k)z_i^k - \varepsilon_k(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)$

The following matrix notation enables succinct algebraic manipulation. Let $\mathbf{X}, \mathbf{Z} \in R^{n\times n}$ and $\mathbf{S}, \mathbf{M} \in R^{r\times r}$ be diagonal matrices defined as,

$$\begin{aligned} X_{ii} &= (x_i^k - \hat{l}_i^k)(\hat{u}_i^k - x_i^k), \quad X_{ij} = 0, i \neq j \\ Z_{ii} &= \left|(\hat{u}_i^k + \hat{l}_i^k - 2x_i^k)z_i^k\right| + 2\varepsilon_k, \quad Z_{ij} = 0, i \neq j \end{aligned} \tag{4.5.24}$$

$$\begin{aligned} S_{ii} &= s_i^k, \quad S_{ij} = 0, \quad i \neq j \\ M_{ii} &= \mu_i^k, \; M_{ij} = 0, \; i \neq j \end{aligned} \tag{4.5.25}$$

This notation allows us to rewrite equations (4.5.19-4.5.23) as:

$$\begin{bmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix}\begin{Bmatrix} \Delta\mathbf{z} \\ \Delta\mu \end{Bmatrix} + \begin{bmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix}\begin{Bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \end{Bmatrix} = \begin{Bmatrix} -\mathbf{v} \\ -\mathbf{w} \end{Bmatrix} \tag{4.5.26}$$

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{C} & \mathbf{I} \end{bmatrix}\begin{Bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \end{Bmatrix} = \mathbf{0} \tag{4.5.27}$$

$$\begin{Bmatrix}\Delta z\\ \Delta\mu\end{Bmatrix}=\begin{bmatrix}A^t & C^t\\ 0 & I\end{bmatrix}\begin{Bmatrix}\Delta\lambda\\ \Delta\mu\end{Bmatrix} \tag{4.5.28}$$

The above linear simultaneous equations can be solved to obtain the stepsize Δx, Δz, Δs, $\Delta\lambda$ and $\Delta\mu$. We make use of the special structure of these equations to obtain the solution efficiently as follows. Premultiplying equation (4.5.26) by the matrix

$\begin{bmatrix}A & 0\\ C & I\end{bmatrix}\begin{bmatrix}Z & 0\\ 0 & M\end{bmatrix}^{-1}$, and substituting equation (4.5.27) and (4.5.28), we get,

$$\begin{bmatrix}AZ^{-1}XA^t & AZ^{-1}XC^t\\ CZ^{-1}XA^t & CZ^{-1}XC^t+M^{-1}S\end{bmatrix}\begin{Bmatrix}\Delta\lambda\\ \Delta\mu\end{Bmatrix}=\begin{Bmatrix}-AZ^{-1}v\\ -CZ^{-1}v-M^{-1}w\end{Bmatrix} \tag{4.5.29}$$

$$\Delta z=A^t\Delta\lambda+C^t\Delta\mu \tag{4.5.30}$$

$$\Delta x=-Z^{-1}X\Delta z-Z^{-1}v \tag{4.5.31}$$

$$\Delta s=-M^{-1}S\Delta\mu-M^{-1}w \tag{4.5.32}$$

Equations (4.5.29) are (m+r) linear simultaneous equations and they can be solved using standard linear equation solvers such as LU decomposition followed by back substitution. The values of $\Delta\lambda$ and $\Delta\mu$ can then be used in equations (4.5.30), (4.5.31) and (4.5.32) to obtain Δz, Δx and Δs respectively.

We need a feasible starting point to minimize the sub-problem (P_k) described in equations (4.5.5)-(4.5.9). Unlike in the linearly constrained problems described in section 4.3, the new solution obtained after an iteration need not be a feasible solution of the sub-problem generated in the next iteration when the constraints are nonlinear. Hence, at each iteration we need to check whether the current solution is feasible and if its not we need to search for the closest feasible point. We do this using the method that was described in section 4.4, with slight modifications to account for the inequality constraints and the slack variables. Firstly, the current point is projected on to the hyper plane represented by the equality constraints. Using equations 4.4.1, we obtain the projected point $\tilde{x}^k$ as

$$\tilde{x}^k=x^k-\nabla h(x_k)(\nabla h(x_k)^t\nabla h(x_k))^{-1}h(x_k) \tag{4.5.33}$$

If the projected point does not satisfy all the inequality constraints, then we reset the move limits and the slack variables and move in the direction of the analytical center of the polyhedra defined by the move limits and slack variables. The analytical center for this polyhedra is obtained by minimizing the following function,

$$P_b\text{: Min } F_b,\ \mathbf{x} \in S_b \tag{4.5.34}$$

$$F_b(\mathbf{x}) = -\varepsilon_k\left(\sum_{i=1}^{n}\ln(x_i - \hat{l}_i^{\,k}) + \sum_{i=1}^{n}\ln(\hat{u}_i^{\,k} - x_i) + \sum_{i=1}^{r}\ln(s_i)\right) \tag{4.5.35}$$

$$S_b = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b}\ ;\ \mathbf{Cx} + \mathbf{s} = \mathbf{d}\ ;\ \hat{\mathbf{l}}^k < \mathbf{x} < \hat{\mathbf{u}}^k\ ;\ \mathbf{0} < \mathbf{s}\} \tag{4.5.36}$$

A few iterations to minimize the above optimization problem, yields a feasible point for the original optimization problem. The move limits are reset each iteration using equation (4.4.3) and the slack variable are initialized to a small positive value.

In section 4.4.3, we described a criteria for resetting the move limits at each iteration. The criteria was found to work very efficiently for linearly constrained problems including the shape and topology optimization problem where the number of variables were very large. Here we propose an alternate criterion for resetting the move limits. This criteria eliminates the dependence of the descent vector on the parameter ε_k. The move limits are updated as follows.

$$a_i = \frac{\mathbf{z}^t \cdot \mathbf{z}}{n \cdot z_i} \tag{4.5.37}$$

$$\begin{aligned} \hat{l}_i^k &= \max\{x_i - a_i,\ l_i\} \\ \hat{u}_i^k &= \min\{x_i + a_i,\ u_i\} \end{aligned},\ i = 1,\ldots,n \tag{4.5.38}$$

Note that the move limits on each variable are set such that they are equidistant from the current value of the variable, except when the variable is very close to the side constraints. When the move limits are equidistant, the descent direction generated at each iteration is not dependent on ε_k as can be verified by equation (4.3.12). Furthermore, this update criteria decreases the distance between the bounds, $2a_i$, as the square of the Kuhn-Tucker vector $\mathbf{z}^t.\mathbf{z}$ decreases. At the optimal solution, when the Kuhn-Tucker vector $\mathbf{z}$ vanishes, the upper and lower move limits converge on to the optimal values of the variables. The inverse relation between a_i and z_i in equation (4.5.37) ensures that the subproblem is properly rescaled during the Newton iterations. We also obtain the Lagrange Multipliers as by-products during the solution process provided none of the side constraints of the original problem become active. In fact if we are interested in obtaining the Lagrange Multipliers we should treat all the side constraints the same as other linear constraints.

The step size suggested by the Newton iteration for the sub-problem depends inversely on ε_k. As the variable approaches the optimal value and the upper and lower move limits converge towards each other, ε_k should tend towards zero so that step sizes do not become too small. We have used the following rule to set the value of ε_k.

$$\varepsilon_k = \frac{g_k}{2n}, \text{ where } g_k = \mathbf{z}^t(\mathbf{x} - \hat{\mathbf{x}}) \tag{4.5.39}$$

g_k is the duality gap between the linear program $\min \mathbf{c}^t\mathbf{x}$, $\mathbf{x} \in S_k$ and its dual $\max q(\lambda)$, $\mu \geq 0$, where

$$S_k = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b};\ \mathbf{Cx} + \mathbf{s} = \mathbf{d};\ \hat{l}_i^k \leq x \leq \hat{u}_i^k; s_j \geq 0\},\ i=1,..n \text{ and } j=1,..r \tag{4.5.40}$$

$$\begin{aligned} q(\lambda) &= \underset{l \leq x \leq u}{\text{infimum}} \{\mathbf{c}^t\mathbf{x} + \lambda^t(\mathbf{Ax} - \mathbf{b}) + \mu^t(\mathbf{Cx} + \mathbf{s} - \mathbf{d})\} \\ &= (\mathbf{c}^t + \lambda^t\mathbf{A} + \mu^t\mathbf{C})\hat{\mathbf{x}} - \lambda^t\mathbf{b} - \mu^t\mathbf{d} \\ &= \mathbf{z}^t\hat{\mathbf{x}} - \lambda^t\mathbf{b} - \mu^t\mathbf{d} \end{aligned} \tag{4.5.41}$$

$$\mathbf{z} = (\mathbf{c} + \mathbf{A}^t\lambda + \mathbf{C}^t\mu) \in R^n \text{ and } \hat{x}_i = \begin{cases} l_i & \text{if } z_i > 0 \\ u_i & \text{otherwise} \end{cases} \tag{4.5.42}$$

The duality gap decreases when $\mathbf{z}$ becomes very small and when the upper and lower move limits of the variables approach each other.

4.6. Summary

This chapter described a sequential approximate optimization technique for nonlinear programming that locally approximates the objective function linearly and sets move limits on the variables using logarithmic barrier functions. This technique, like other sequential linearization techniques, does not require the evaluation of the hessian of the objective function. In addition, computation per iteration is reduced by not solving the sub-problems completely. The move limits of the sub-problem are readjusted each iteration so that the upper and lower limits move toward each other as the variables approach their optimal values. These move limits reduce or eliminate the need for step-size reduction at each iteration.

The algorithm requires the user to provide a starting point or a set of side constraints within which the optimal point lies. An initial feasible point is found by projecting the starting point on to the hyper plane represented by the equality constraints and moving along this hyperplane iteratively towards the analytical center of the feasible polyhedra until

a feasible point is found. The algorithm can handle inequality constraints also by converting them into equality constraints using slack variables. The Lagrange Multipliers can be obtained as a by-product of constrained minimization if the side constraints are also converted to equality constraints using slack variables.

Preliminary tests indicate that the algorithm works fairly well for unconstrained as well as constrained problems. Examples to test the algorithm as well as its application to shape and topology optimization are included in chapter 5. The algorithm seems to be particularly suited for structural optimization problems as is illustrated in the shape and topology optimization examples.

5 Implementation

5.1. Overview

In this chapter, a brief description is given of the implementation of the shape and topology optimization as well as the sequential optimization (MBSLP) algorithm. The software implementation consists of two parts. The first part consists of a pre- and post-processor, that provides a graphical interface to define the design problem and to display the results respectively. The shape and topology optimization algorithm is implemented in a software that will be referred to as OPT. The pre-processor writes out the data associated with the problem definition in a file that serves as input to OPT. OPT computes the optimal shape and topology in the shape density function representation. The results generated by OPT are written into a file that serves as input to the post-processor that displays the result graphically.

In section 5.2, the shape and topology optimization algorithm is described. This section also describes the implementation of the pre-processor and the post-processor. The design problem is specified in the preprocessor which generates the finite element mesh and related data to describe the design model. The results are displayed by the post-processor. The techniques / algorithms employed in both the pre- and post-processors are briefly described and the related references cited. The implementation of the sequential approximate optimization algorithm is explained in section 5.3. Finally, in section 5.4 we summarize the chapter.

5.2. Shape and Topology Optimization Algorithm

5.2.1. Design problem specification (Pre-processor)

The design problem specification for shape and topology synthesis consists of specifying the following information:

1) A feasible domain (or design domain) within which the final optimal structure should fit. This region specifies the maximum volume the structure can occupy and it excludes regions where other parts must fit. This enables obstacle avoidance and prevents interference with other component in the assembly in which the component being designed must fit.

2) The loads to be carried by the structure (design loads) and their location.

3) Support conditions on the structure, that is the boundary conditions.

4) The material properties of the structure.

An example for an arbitrary planar structural design is illustrated in figure 5.1. The preprocessor provides a graphical interface to interactively specify the design problem. The feasible domain is specified by defining its boundaries in the plane. It is possible to specify non convex regions with internal holes. The external boundary is specified counter-clockwise and the internal boundaries are specified clockwise. This enables the algorithms to distinguish between the interior and the exterior of the domain. This feasible domain is taken as the initial guess for the geometry. Nodes are placed along the boundaries of the feasible domain at user specified intervals. The design loads are specified on the appropriate nodes along the boundary. The type of support provided to the structure can also be specified by defining boundary conditions on the nodes along the boundaries that have external supports.

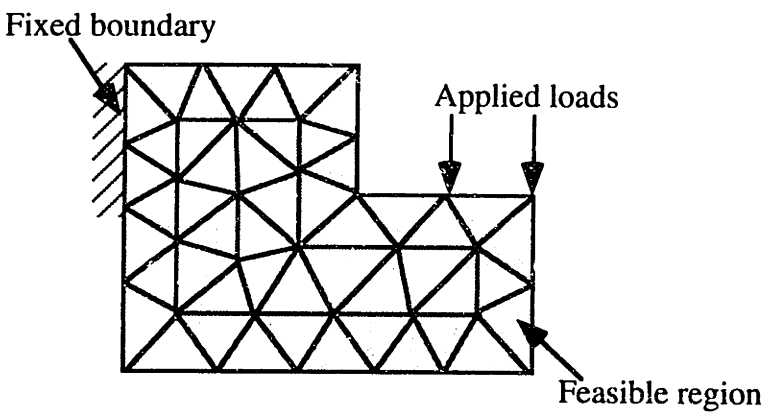


Figure 5.1. Design problem specification

As explained in chapter 3, structural behavior is analyzed using the finite element method. The feasible domain must therefore be divided into a finite element mesh. In this implementation we have used triangular elements. The shape is represented using the shape density function as was explained in chapter 3. This density function is also represented by

piece-wise linear interpolation over this triangular mesh. A mesh generation algorithm was implemented that takes the feasible domain as the input and divides the region into a triangular mesh. The current implementation requires the users to specify the boundary nodes as well as some of the internal nodes if necessary. The mesh generation algorithm automatically adds some internal nodes and then connects all the nodes together to form a triangular mesh using a triangulation algorithm. For finite element applications, the ideal triangulation consists of triangles that are as close to equilateral as possible.

A Delaunay triangulation algorithm was implemented in the preprocessor that automatically connects a given set of points into triangles. The details of Delaunay triangulation, its properties and an algorithm for generating them are briefly described below.

Of all possible triangulations for a given set of points in a plane, the one that consists of triangles closest to equilateral triangles is Delaunay triangulation. In finite element mesh generation applications, Delaunay triangulation of nodal points has come to be accepted as the best triangulation since most triangular element interpolations involve mapping to a parametric space where the triangles are assumed to be equilateral. If the actual element shape is close to equilateral then distortion in the mapping is reduced. Another desirable property of Delaunay triangulation is that the triangles never overlap, and therefore, no additional checking is necessary to ensure non-overlapping elements. Delaunay triangulation is unique to a given set of points except in a degenerate case. A formal definition of Delaunay triangulation and the related concept of Voronoi polygons is given below.

Voronoi polygons: Let $\mathbf{P} = \{ P_i, i=1,N \}$ be a set of N distinct points in a plane ($\mathbf{R}^2$), then, Voronoi polygons have been defined [Ho-Le_88],[Sloan_87], as the set of polygons, $\mathbf{V}=\{V_i,i=1,N \}$, where,

$$V_i = \{ x \in R^2 : | x - P_i | < | x-P_j | \; \forall \, j \neq i \}$$

and $|\cdot|$ denotes the Euclidean distance norm. Hence, all the points in the i^{th} Voronoi polygon is closer to its generating point P_i than any other point in the given set $\mathbf{P}$.

Delaunay triangulation: The set of triangles formed by connecting the generating point of neighboring Voronoi polygons is called the delaunay triangulation. In general, a vertex of a polygon is shared by two other neighboring polygons. Hence, connecting the generating points of neighboring polygons yield triangles.

The two most important properties of Delaunay triangles are listed below. These properties have been used as an alternative definition of Delaunay triangles and have been used in triangulation algorithms.

Max-Min angle criteria: For any pair of triangles as shown in figure 5.2, the minimum angle for the two cases I and II are compared and if $\alpha 1 < \alpha 2$, then case II represents Delaunay triangulation.

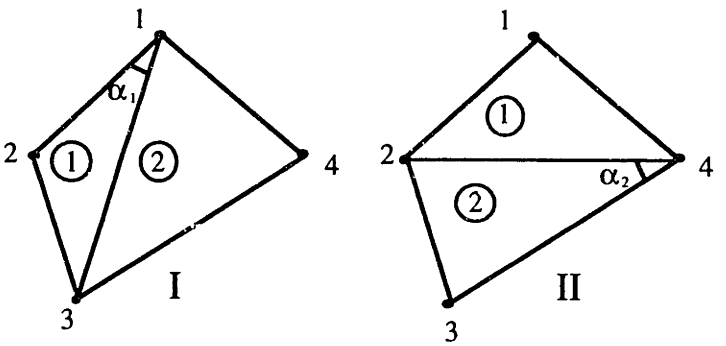


Figure 5.2. Max-Min angle criterion

Circle criterion: In the Delaunay triangulation of a given set of point, the circumcircles of the triangles do not contain any other point of the given set. Hence, for the pair of triangles shown in figure 5.3, if the circumcircle of the triangle ijk contains the point p, then the triangulation is not Delaunay. The diagonal ik has to be swapped to form Delaunay triangles.

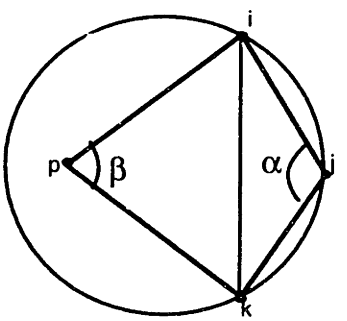


Figure 5.3. Circle criterion

This criterion is computationally the most economical means of constructing Delaunay triangles. This criterion may be checked easily, using the property that if the sum of the angles $\alpha+\beta > \pi$, then point p lies inside the circumcircle of the triangle ijk. If $\alpha+\beta = \pi$, then the point p lies on the circumcircle of the triangle ijk. This is the degenerate case, when Delaunay triangulation is not unique. Both the possible choice of triangulation are equally acceptable.

We have used Lawson's algorithm [Lawson_77] to construct the Delaunay triangulation. Using the criteria described above, any set of four points can be connected optimally by swapping the diagonal of the quadrilateral if necessary, as shown in figure 5.3. This makes the triangulation locally optimal (Delaunay). Lawson shows that repeated application of this local optimization to pairs of triangles, finally terminates in the global optimal triangulation in a finite number of steps. This is a consequence of the fact that Delaunay triangulation is unique and that local optimization by swapping diagonals does not affect any triangles other than the two involved in the swapping. An efficient implementation for this algorithm is described by Sloan [Sloan_87].

When the domain is non-convex and it contains internal boundaries, the above procedure of triangulation cannot be strictly applied. The initial Delaunay triangulation yields the convex hull of the given set of points. Therefore, for non-convex geometries some external elements are generated that have to be identified and removed. Also it is essential that in the initial triangulation no edge of a triangle, should cross a internal boundary. To ensure this, swapping an edge is prevented during local optimization, if the edge lies along a boundary. Similarly, swapping is enforced if doing so makes the edge lie along the boundary. In our implementation the nodes on the boundary are ordered sequentially such that the nodes on the external boundary are numbered counter-clockwise and the nodes on the internal boundary are numbered clockwise. As one moves along the boundaries in these specified directions, triangles that lie on the right side of the boundaries are the external triangles. These triangles are detected and removed.

Once the mesh is generated it is beneficial to renumber the nodes to reduce the bandwidth of the stiffness matrix. This reduces the computation per finite element analysis. Since the optimization algorithm performs the analysis for each iteration, bandwidth minimization yields significant improvement in performance. The preprocessor includes a bandwidth minimization function that renumbers the nodes optimally. The algorithm used for bandwidth minimization is described in [Gibbs_76]. The design specification and the data generated by the preprocessor are then written to a file that serves as the input to OPT. The input file to OPT contains the following information :

1) The number of nodes in the mesh.
2) The nodal coordinates of all the nodes.
3) The number of elements.
4) The element connectivity data (the node numbers of the element vertices).
5) The fixed nodes (boundary conditions).

6) The applied loads (the load vectors and the node numbers on which they are applied) and

7) the material properties (Young's modulus and Poisson's ratio).

Figure 5.4 shows an example model created using the preprocessor. The nodes are displayed by the small squares and the fixed nodes are marked by slightly larger squares. The applied load is shown by the arrow.

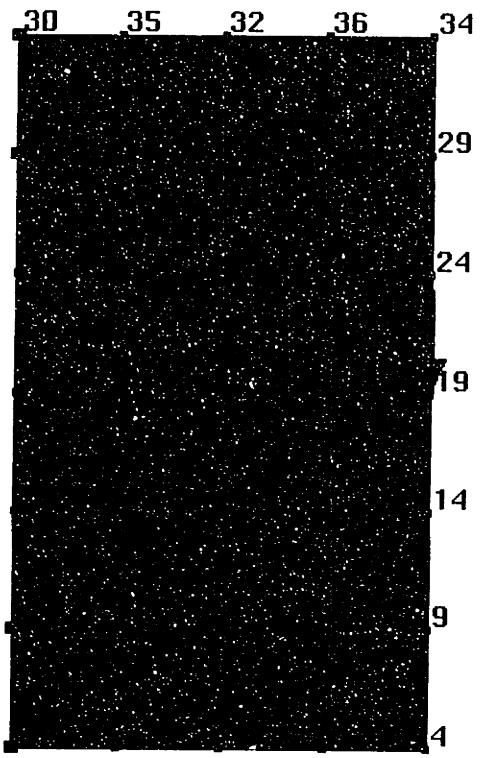


Figure 5.4. A model generated by the preprocessor

5.2.2. Description of the algorithm

In chapter 3 we described the ideas and concepts behind the shape and topology optimization method. The algorithms implemented in OPT is described in this section. Figure 5.5 illustrates the algorithm for implementing the linear approximate material property-density relation. The design problem specification in the form of the feasible region, loads and boundary conditions is read in from the input file generated by the preprocessor. Material is removed in small steps. The user can specify the percentage of material to be removed in each step as well as the number of steps.

The feasible region is take as the initial guess for the shape. The shape density function is therefore, set equal to unity everywhere in the feasible domain. The material is removed iteratively as shown by the loop in flow chart (figure 5.5). In each iteration the mass of the structure (W) is computed by integrating the density function over the feasible region (see equation 3.3.4). The gradient of the mass with respect to the nodal values of density is also computed at the beginning of each iteration. The constraint on the mass of the structure is a linear constraint so that the gradient of the mass is a constant vector (represented as **A**). The mass of the structure can be expressed in terms of this gradient vector as,

$$W(\phi) = \int_{\Omega} \phi d\Omega = \mathbf{A}\phi_i \tag{5.2.1}$$

where, $\mathbf{A} = \nabla W$ and ϕ_i are the nodal values of the density function. The constraint is specified such that the mass of the optimized geometry is reduced at each iteration by the amount specified by the user. The constraint on the mass is stated as an equality constraint as

$$W(\phi) = \mathbf{A}\phi_i = (1-r)W_o \tag{5.2.2}$$

where, r is the percentage of mass to be removed at each iteration and Wo is the current mass of the structure.

The optimization algorithm is used to solve for the optimal geometry. The nodal value of the density functions are treated as the design variables. This algorithm is described in detail in section 5.3. The algorithm expects the user to provide routines that evaluate the objective function and the gradient of the objective function. In our implementation the objective function (that is the strain energy L(**u**) of the structure, defined in equation 3.3.3) is evaluated using the finite element method. The gradient of the objective is evaluated using the sensitivity analysis techniques described in section 3.6. The algorithm also expects the user to define the constraints as well as the gradients of the constraints. Since the constraint on the mass is linear, we need to evaluate its gradient only once each iteration to define the constraint (as in equation 5.2.2). The side constraints on the design variables are set as

$$\phi_{th} - 0.01 \le \phi_i \le 1.0 \tag{5.2.3}$$

The threshold value ϕ_{th} was set at values higher than 0.5 to ensure that the material property coefficients do not become negative in the allowed range of density variation. Once the optimal design is found for the given constraint on mass, the geometry is updated

by removing elements for which all three nodal density values lie below the threshold value. This updated geometry is used as the starting geometry for the next iteration. The iterations or the material removal steps are carried out as many times as the user specifies. Some of the results obtained by this algorithm are illustrated in chapter 6. It is noted that the density function values transition very sharply from the lowest value to the highest value (equation 5.2.3) at the boundary of the solid. As a result we have clearly defined boundaries. However, since the geometry is updated at each iteration the results seem to depend on the mesh density and quality. This is seen in some examples included in chapter 6.

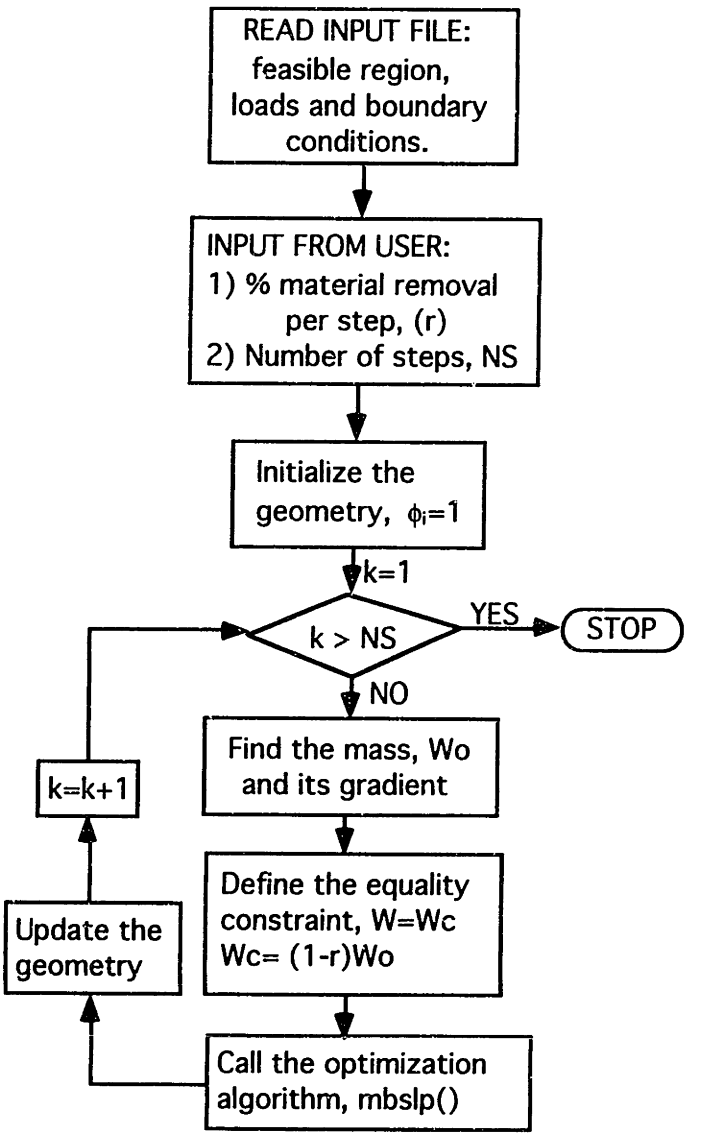


Figure 5.5. OPT algorithm (Material removal in steps)

In section 3.4.2 we described quadratic and higher order material property density relations. As noted in chapter 3, for this case the material property coefficients are positive

for all values of density in the interval [0,1]. Therefore, we can set the threshold value to zero. Regions where the density value is reduce to zero do not contribute to the stiffness of the structure. Therefore, the material can be removed in one step and there is no need to update the geometry by removing elements. The algorithm is illustrated in figure 5.6. The design specification is read in from the file generated by the preprocessor. The user can specify the percentage of material to be removed. Rather than remove the material in many small steps, this algorithm removes the specified percentage in one step. The optimal designs thus obtained may have large regions where the density has intermediate values. In order to get clearly defined boundaries and fully dense optimal shapes, it is beneficial to add a penalty function to the objective function that penalizes intermediate densities. The penalty function is described in section 3.4.2 (equation 3.4.6). The optimal shape obtained without penalty on intermediate values of density is taken as the initial guess for subsequent optimizations with penalty imposed on intermediate densities. The penalty constant c_p in equation (3.4.6) is increased in steps. The user can specify the number of such steps.

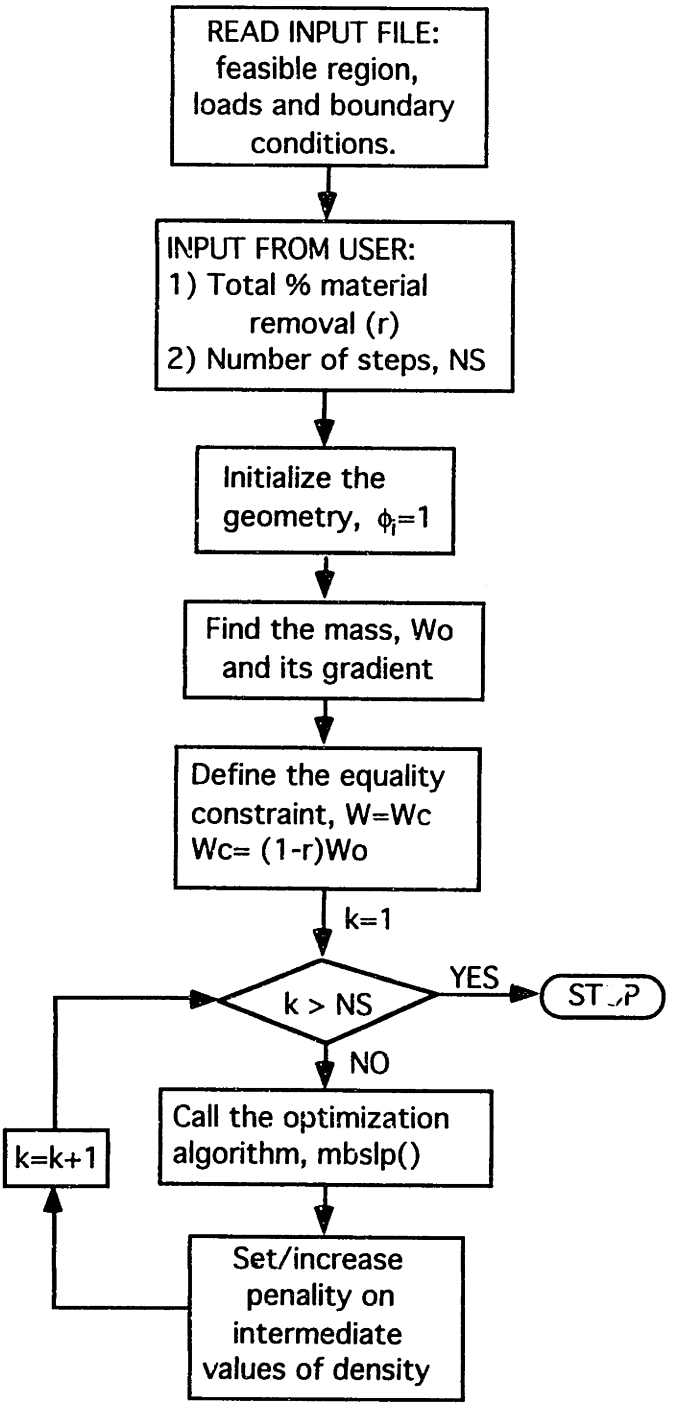


Figure 5.6. OPT algorithm

In the first step, the initial guess of the optimal geometry is set to be the same as the feasible region. The nodal density values are set to unity in the entire feasible region. The constraint on the mass of the structure is set such that the mass of the optimal structure is less than the initial guess (the entire feasible region) by the percentage (r) specified by the user. This constraint is set as in equation (5.2.2). The optimization is carried out using the algorithm. The side constraints on the design variables is set as

$$\phi_{th} \leq \phi_i \leq 1.0 \tag{5.2.4}$$

The threshold value was set to very low values (0.01-0.05) for this algorithm since it uses quadratic and higher order approximations. The threshold value was not set to zero to avoid any singularities that occasionally arise. The optimal geometry thus obtained is taken as the initial guess for subsequent iterations where a penalty is added to the objective function to remove intermediate densities. The penalty constant is increased each iteration by a fixed factor. In this implementation, we set the penalty constant as c_p=10 and increase it each iteration by factor of 10.

5.2.3. Displaying the results (Post-processor)

At the end of the optimization process, the optimal shape as well as the analysis results from the finite element analysis program are written out in a file. These results can be graphically visualized using the post processor. Since shape is represented using the shape density function, it is represented in OPT as a list of nodal density values corresponding to all the nodes in the finite element mesh. In order to visualize this shape the post processor can display the fringes of the shape density function in the feasible domain. In figure 5.7. a simple example is shown. The model has very few nodes and elements. The rectangular region is the feasible region. The fringes of the shape density function are plotted such that white corresponds to fully dense material ($0.9 \leq \phi \leq 1$), while black represents the lowest possible density allowed by the side constraints. Intermediate values of the density function are represented by 10 levels of gray.

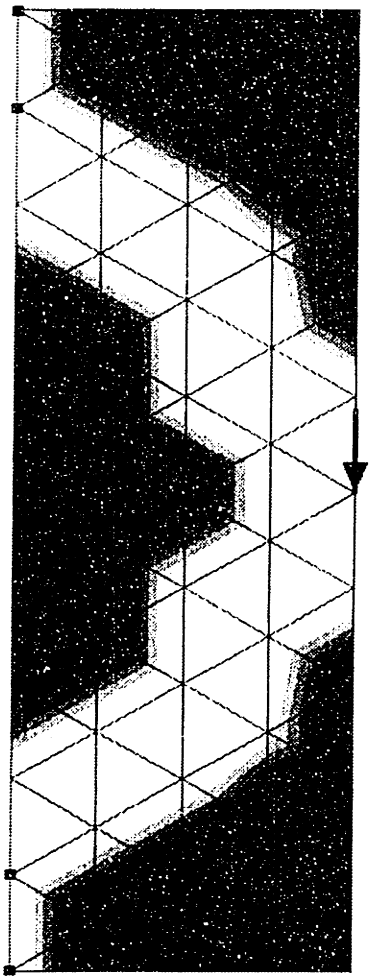


Figure 5.7. Fringes of shape density function

A contour corresponding to the threshold value of the density is then interpreted as the boundary of the shape. Regions with density values larger than the threshold value are considered to be the interior of the structure. In the example illustrated in figure 5.7, the threshold value was set at 0.01.

5.3. MBSLP Algorithm

In chapter 4, a sequential approximate optimization technique is proposed which we refer to as the moving barrier sequential linear programming algorithm. In this section, the implementation of the algorithm is described. This algorithm was implemented in C language and executed in a UNIX environment.

The optimization algorithm is implemented in a function called mbslp(). The user has to define three functions:

1) A main function that allocates memory for the data structure *opt*, initializes the variables and calls the function mbslp() to solve for the optimum.

2) A function that evaluates the objective function and the constraint functions.

3) A function that evaluates the gradients of the objective and constraint functions.

The software assumes that the objective and constraint functions are defined in a function called eval_function(opt,x). The arguments to this function are a pointer to the data structure (*opt*) that contains the variables and the arrays associated with the optimization algorithm and an array (x) that contains the value of the variables at which the functions are to be evaluated. This function returns the value of the objective function and stores the values of the constraint functions in the structure *opt*. Similarly, the gradients of the objective and constraint functions are assumed to be evaluated by a function named eval_gradient(opt,x). The optimization data structure, *opt*, consists of the following information:

1) The number of variables, the number of equality constraints and the number of inequality constraints.

2) An array that holds the values of the variables.

3) The values of the objective function and each constraint function evaluated at the current value of the variables.

4) The gradient vector for the objective function and each of the constraint functions.

5) Two arrays that store the lower and upper bounds on the variables which are also referred to as the side constraints.

The users must specify an initial value for the variables and also the side constraints on the variables in their main function. The algorithm implemented in the function mbslp() is schematically represented in figure 5.8.

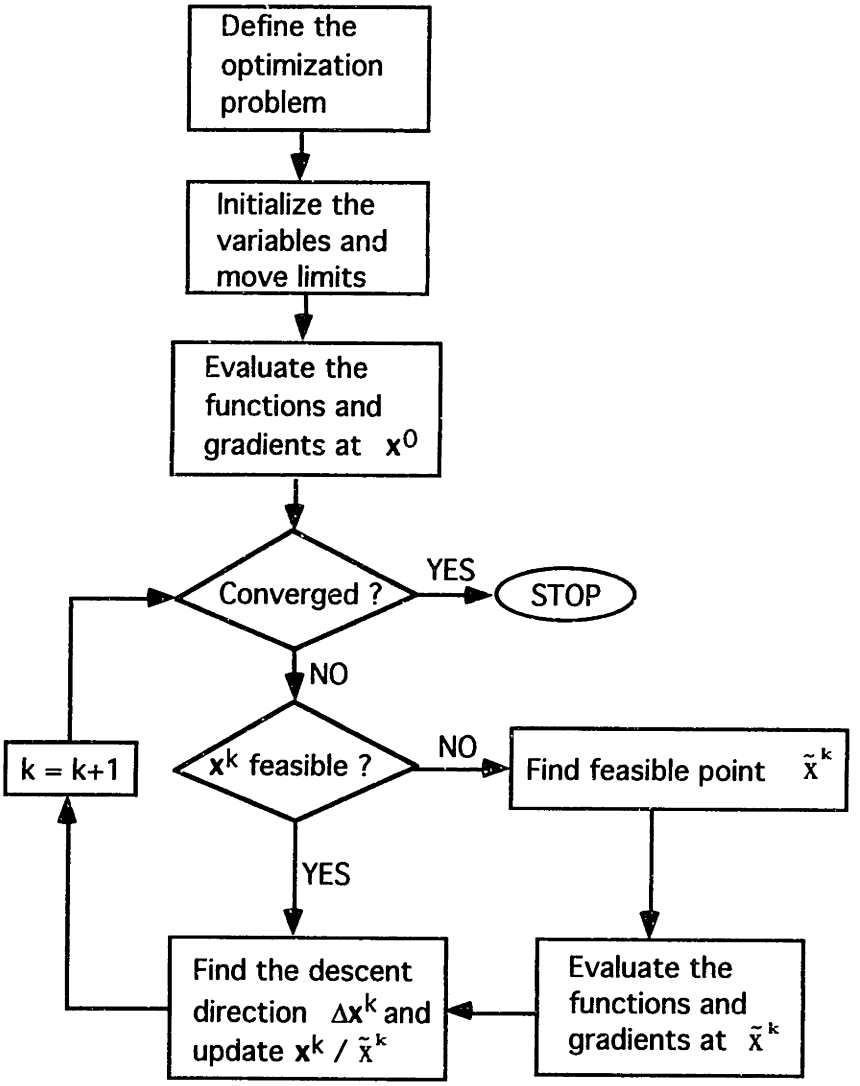


Figure 5.8. MBSLP algorithm

The algorithm initializes the move limits to the lower and upper bounds. The functions and gradients of the objective and constraints are evaluated at the initial guess for the variables. A simple convergence criteria is used to check whether the algorithm has converged. The descent direction is found iteratively until the variables satisfy this convergence criteria. At each iteration the current value of the variables are tested to see if it represents a feasible point, that is whether all the constraint equations are satisfied. If any of the constraints are violated, the nearest feasible point is found using the method described in section 4.4. The implementation of the algorithm for finding a feasible point is described later in this section. The functions and gradients are re-evaluated at the new feasible point. A descent direction is found using equations (4.5.29)-(4.5.32). The variables are updated using equations (4.3.21) and (4.3.22) after selecting a step size using Armijo's criteria described in equation (4.3.24). The details of the algorithm to find a

descent direction is described later in this section. The iterations are stopped when the convergence criterion is met or when the number of iterations exceed the maximum permissible limit.

In this implementation, we have used duality gap as a convergence criterion. It is defined as

$$g_k = \mathbf{z}^t(\mathbf{x} - \hat{\mathbf{x}}) \tag{5.3.1}$$

g_k is the duality gap between the linear program $\min \mathbf{c}^t\mathbf{x}$, $\mathbf{x} \in S_k$ and its dual $\max q(\lambda)$, $\mu \geq 0$ defined in equations (4.5.40) and (4.5.41). When the duality gap g_k is smaller than a preset tolerance the iterations are terminated. At each iteration, the Kuhn-Tucker vector $\mathbf{z}$ gets smaller and the move limits converge towards each other and as a result the duality gap becomes smaller.

The descent direction is found at each iteration by minimizing the subproblem (P_k), defined in equations (4.5.5)-(4.5.9), using Newton's method. The subproblem and the method to find a descent direction are described in section 4.3 for linearly constrained problem and in section 4.5 for the general case of nonlinear equality and inequality constraints. The algorithm for computing a descent direction is illustrated in figure 5.9.

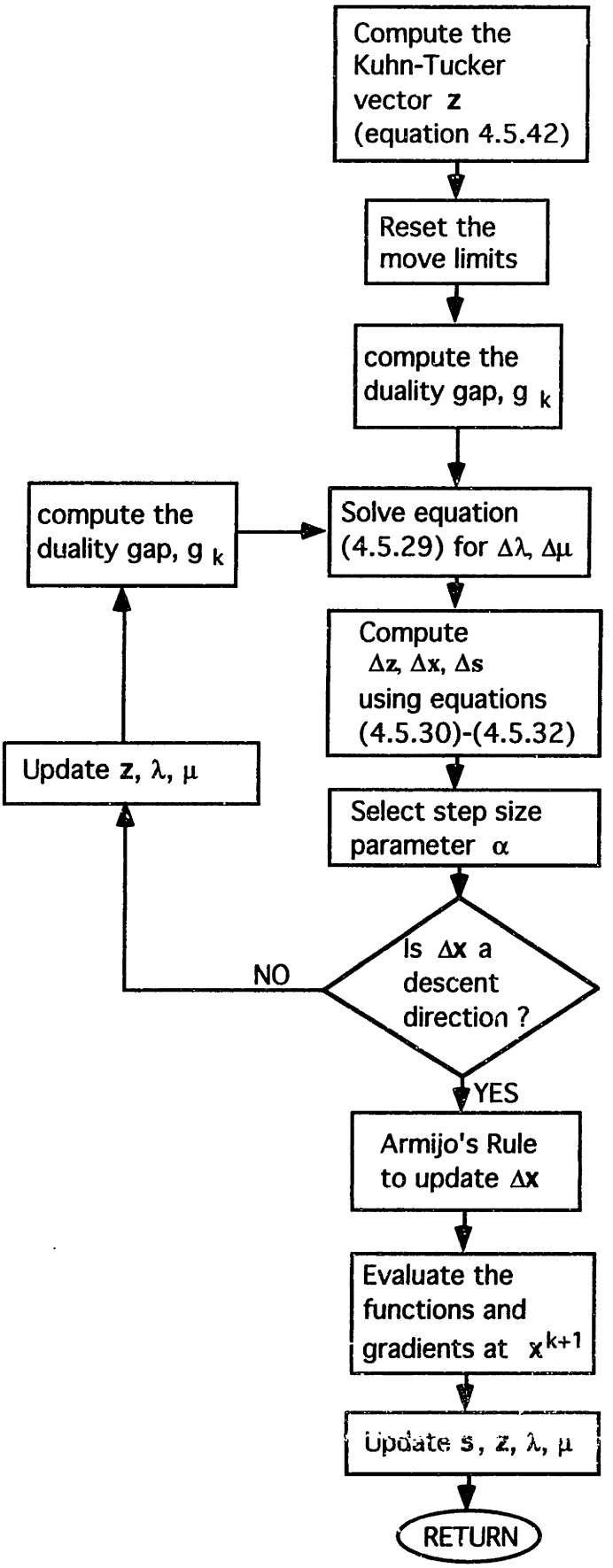


Figure 5.9. Algorithm to compute a descent direction

This algorithm is implemented in a function called primdual(). The input arguments to this function are the optimization data structure *opt*, the maximum permissible number of iterations and the arrays containing the move limits on the variables. The algorithm first computes the Kuhn-Tucker vector, $\mathbf{z} = (\mathbf{c} + \mathbf{A}^t\lambda + \mathbf{C}^t\mu)$, using the current values of the Lagrange multipliers. The Lagrange multipliers corresponding to the linear constraints , λ, are initialized to unity, while those for nonlinear constraints, μ, are initialized to a small positive number at the first iteration. The Lagrange multipliers are updated each iteration and these new values are used in the next iteration to recompute the **z** vector. Thereafter, the move limits are updated using one of the criteria defined in chapter 4.

The new duality gap is computed using the **z** values and the move limits computed in this iteration. Equation (4.5.39) are used to compute the new values for ε_k. Now using the current values of the arrays **x**, **s**, **z**, λ and μ the coefficient matrix on the left hand side of equation (4.5.29) as well as the right hand side vector can be computed. It can be verified easily that the matrix is symmetric. Therefore only the upper triangular part of the matrix need be computed. Also note that the Z and X matrix are diagonal. Making use of this fact, we compute the sub-matrices in equation (4.5.29) as

$$[\mathbf{A}\mathbf{Z}^{-1}\mathbf{X}\mathbf{A}^t]_{ij} = \sum_{k=1}^{n} A_{ik}\frac{X_{kk}}{Z_{kk}}A_{kj} \tag{5.3.2}$$

$$[\mathbf{A}\mathbf{Z}^{-1}\mathbf{X}\mathbf{C}^t]_{ij} = [\mathbf{C}\mathbf{Z}^{-1}\mathbf{X}\mathbf{A}^t]_{ij} = \sum_{k=1}^{n} A_{ik}\frac{X_{kk}}{Z_{kk}}C_{kj} \tag{5.3.3}$$

$$[\mathbf{C}\mathbf{Z}^{-1}\mathbf{X}\mathbf{C}^t + \mathbf{M}^{-1}\mathbf{S}]_{ij} = \sum_{k=1}^{n} C_{ik}\frac{X_{kk}}{Z_{kk}}C_{kj} + \frac{S_k}{\mu_k}\delta_{ij} \tag{5.3.4}$$

Similarly, we compute the two sub vectors of the right hand side vector as

$$[\mathbf{A}\mathbf{Z}^{-1}\mathbf{v}]_i = \sum_{k=1}^{n} A_{ik}\frac{v_k}{Z_{kk}} \tag{5.3.5}$$

$$[\mathbf{C}\mathbf{Z}^{-1}\mathbf{v} + \mathbf{M}^{-1}\mathbf{w}]_i = \sum_{k=1}^{n} C_{ik}\frac{v_k}{Z_{kk}} + \frac{w_i}{\mu_i} \tag{5.3.6}$$

After assembling the left hand side coefficient matrix and the right hand side vector, equations (4.5.29) are solved using standard linear simultaneous equation solvers to obtain the vectors $\Delta\lambda$ and $\Delta\mu$. In our implementation, we use LU decomposition algorithm to solve the equations [Press_88].

The vectors $\Delta\mathbf{z}$, $\Delta\mathbf{x}$, $\Delta\mathbf{s}$ can then be computed using equations (4.5.30)-(4.5.31). The variables are updated as follows,

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{x}^k + \alpha_k \Delta\mathbf{x} \\ \mathbf{z}^{k+1} &= \mathbf{z}^k + \alpha_k \Delta\mathbf{z} \\ \lambda^{k+1} &= \lambda^k + \alpha_k \Delta\lambda\end{aligned} \tag{5.3.7}$$

The step size α_k is selected such that the new values of the variables are feasible, i.e., $\hat{\mathbf{l}}^k < \mathbf{x} < \hat{\mathbf{u}}^k$ and $\mathbf{s} > \mathbf{0}$. The step size parameter α_k may therefore be computed as follows,

$$\alpha_k = \min_{i=1,\ldots,n}\{\tilde{\alpha}_i\}, \text{ where} \tag{5.3.8}$$

$$\tilde{\alpha}_i = \min\left\{\begin{array}{l}\dfrac{\hat{l}_i^k - x_i^k}{\Delta x_i^k}, \text{if } \Delta x_i < 0; \ \dfrac{\hat{u}_i^k - x_i^k}{\Delta x_i^k}, \text{if } \Delta x_i > 0; \\ \dfrac{-\mu_i}{\Delta\mu_i}, \text{if } \Delta\mu_i < 0; \ \dfrac{-s_i}{\Delta s_i}, \text{if } \Delta s_i < 0;\end{array}\right\}, \ i=1,..n \tag{5.3.9}$$

Before updating all the variables using equation (5.3.7), we check whether $\Delta\mathbf{x}$ is a descent direction, that is, if $\mathbf{c}^t\Delta\mathbf{x}<0$. If it is not a descent direction, then we update only $\mathbf{z}$, λ and μ. The duality gap and ε_k are re-evaluated another Newton iteration is carried out for the sub problem. These iterations are continued until a descent direction is found. For unconstrained and linearly constrained problems descent direction is found at the very first iteration. After a descent direction is found, we use Armijo's rule to check whether the step size is excessive. The Armijo rule used in our implementation is given below. We reset the step size as $\alpha_k \approx \beta^{m_k}\alpha_k$, where we chose a fixed scalar β such that $0<\beta<1$ and m_k is the smallest positive integer for which the following relation holds.

$$f(x_k) - f(x_k + \beta^{m_k}\alpha_k\Delta x_k) \geq -\sigma\beta^{m_k}\alpha_k\nabla f(x_k)^t\Delta x_k \tag{5.3.10}$$

The move limits and the corresponding barrier functions limit the step size. Therefore, Armijo's rule is satisfied in most cases and no further step size reduction is required. After updating all the variables, the functions and gradients are evaluated at $\mathbf{x}^{k+1}$. These values are returned to the calling function.

The algorithm for finding a feasible point is illustrated in the figure 5.10. This algorithm is implemented in a function called find_feasible_point(). The current design vector (vector of design variables) and the functions evaluated for this value of the variables are passed to this function.

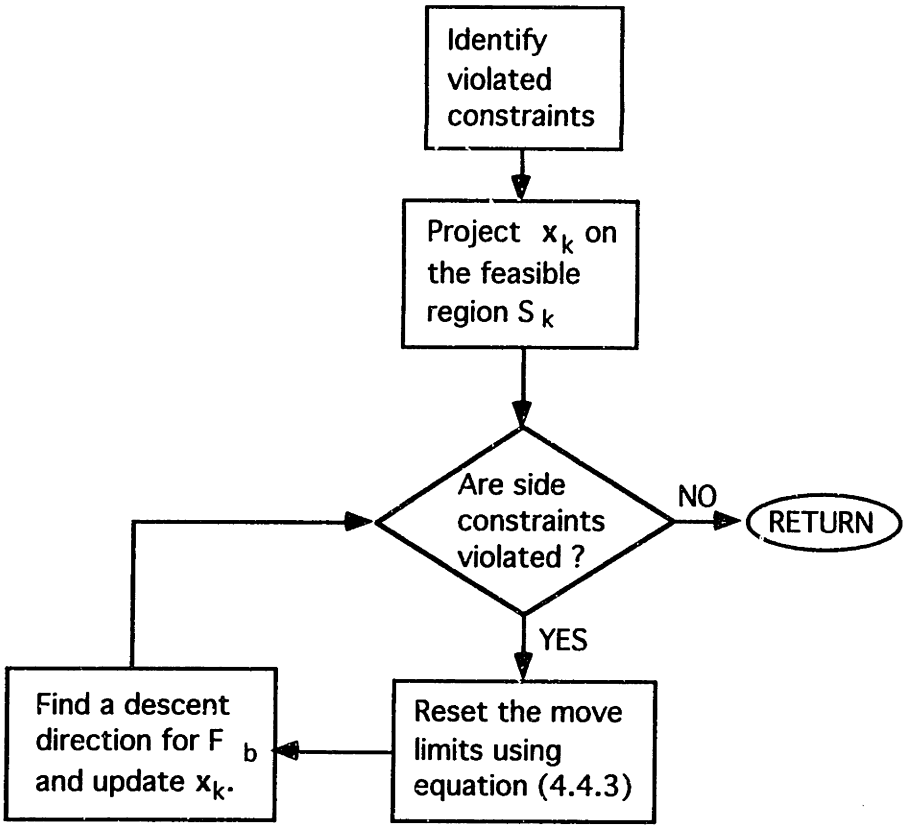


Figure 5.10. Finding a feasible point

The algorithm first identifies the constraints that are violated. Then it projects the current design vector on to the set of feasible points, $S = \{\mathbf{x} \mid \mathbf{Ax=b}, \mathbf{Cx+s=d}\}$ using equations (4.5.33). The projected point may still violate some of the side constraints and/or the move limits. The move limits are reset using equation (4.4.3) such that the projected point is enclosed within the move limits. The minimum of the function F_b defined in equation (4.5.3) is the analytical center of the feasible polyhedra defined in equation (4.5.36). The algorithm for finding the descent direction of the sub problems can also be used to find a descent direction of the function F_b after setting $\mathbf{c} = 0$. The variables are now updated using the descent direction thus obtained. After a few such iterations we obtain a feasible point that satisfies the side constraint as well.

6 Results and Discussions

6.1 Overview

In this chapter the concepts and algorithms described in the previous chapters are applied to examples to illustrate their performance. In section 6.2 the MBSLP algorithm is applied to simple test problems whose solutions are known analytically. To illustrate the generality of the algorithm, we apply it to unconstrained, linearly constrained and nonlinearly constrained problems. However, the algorithm has been extensively tested only for linearly constrained problems. The criteria for resetting the move limits described in section 4.3.4 was found to perform well for linearly constrained problems. For nonlinearly constrained problems it may be possible to design better criteria that take into account the nature of the constraint functions. For the shape and topology optimization problem, the objective function is the strain energy of the structure. This objective function is a highly nonlinear function of the design variables (that is the nodal values of the density function). The constraint on weight is however a linear function of the variables.

In section 6.3 we apply the methods discussed in chapter 3 to design the shape and topology of structures. Many specific examples of planar loading conditions are considered. The merits and demerits of the various material property-density relations proposed in section 3.4 are discussed. The effect of mesh density and the quality of the mesh are also studied. In section 6.4, the chapter is summarized and some the key results are discussed.

6.2 Examples using MBSLP algorithm

In this section, we give examples to illustrate the application of the sequential approximate optimization algorithm to solve nonlinear programming problems. The MBSLP algorithm is applied here mostly to simple test examples that have very few

variables and whose solution are known analytically. In the next section, we apply the algorithm to shape and topology optimization of structures.

Example 6.2.1. Unconstrained Optimization

In this example we consider a simple unconstrained nonlinear program with 2 variables. The optimization problem may be stated as :

$$\min f(x_1,x_2) = \frac{(x_2+y_2)-x_1y_1}{1000.}, \tag{6.2.1}$$

where,

$$y_1(x_1,x_2) = 50000. - 5000.x_1 + 40.x_2 - x_1x_2 - 0.002x_2^2 \text{ and} \tag{6.2.2}$$

$$y_2(x_1,x_2) = 100000. + 2y_1 \tag{6.2.3}$$

Initial points were given as $(x_1,x_2) = (1.,10.)$ and the side constraints on the variables were set as $0. < x_1 < 15.$ and $0. < x_2 < 10000.$ The results are tabulated below.

Table 6.1. Unconstrained optimization (Example 6.2.1)

Iteration	f(x)	x1	x2	$z^tz=c^tc$
1.	-235.04	4.88	3190.80	2540.14
2.	-716.45	9.29	5596.90	8767.80
3.	-786.40	11.06	7815.70	1267.46
4.	-793.60	10.20	7165.98	309.40
5.	-793.89	10.99	7241.18	98.20
6.	-794.94	10.77	7094.08	100.14
7.	-795.52	10.35	7323.2	11.76
8.	-796.05	10.58	7308.75	27.16
9.	-796.07	10.55	7331.17	0.19
13.	-796.070	10.559	7330.951	1.0e-6

The contours of the function $f(x_1,x_2)$ are plotted in figure 4.3. The function has a very ill-conditioned hessian matrix, but our algorithm performs well in this case because the hessian is nearly diagonal (that is, the off-diagonal terms are nearly zero). The algorithms exhibits very fast convergence for this problem as can be seen in table 1. The algorithm took fewer iterations than modified Newton algorithms and fewer function evaluations. The algorithm is not expected to take fewer iterations than other sequential programming

algorithms, however the computation required per iteration will be lower since only one or two Newton steps were required per iteration.

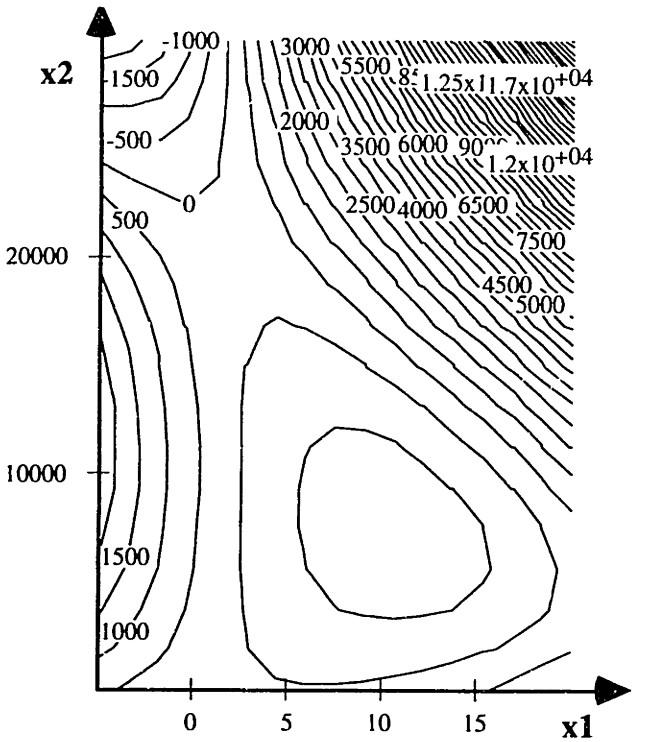


Figure 6.1. Contours of the function $f(x_1,x_2)$ (Example 6.2.1)

Example 6.2.2 Quadratic Program

We consider a quadratic program with 5 variables, subject to three linear equality constraints. This example is problem 53 in [Hock_81].

$$\min f(\mathbf{x}) \qquad (6.2.4)$$
$$f(\mathbf{x}) = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2$$

subject to,

$$h_1(\mathbf{x}) = x_1 + 3x_2 = 0$$
$$h_2(\mathbf{x}) = x_3 + x_4 - 2x_5 = 0 \qquad (6.2.5)$$
$$h_3(\mathbf{x}) = x_1 + 3x_2 = 0$$

The bounds on the variables were set as $-10 \le x_i \le 10.$, i=1,..5. An initial feasible point was found as (-0.46, 0.154, 0.154, 0.154, 0.154) for which the function value was $f(\mathbf{x})$=4.674. The iterations are tabulated below.

Table 6.2. Quadratic program iterations (Example 6.2.2)

Iter.	f(x)	x1	x2	x3	x4	x5	$z^t z$
1.	4.3540	-0.948	0.316	0.356	0.276	0.316	11.36
2.	4.1175	-0.685	0.228	0.549	-0.093	0.228	1.694
3.	4.1149	-0.855	0.285	0.600	-0.030	0.285	0.667
4.	4.0930	-0.765	0.251	0.625	-0.116	0.255	0.152
20.	4.0930	-0.7676	0.2258	0.6279	-0.1163	0.2558	1.0e-6

Example 6.2.3. Nonlinear constraints

The following example (problem 18 in [Hock_81]) has a quadratic objective function of 2 variables. The two inequality constraints are nonlinear.

$$\min f(\mathbf{x}) \qquad (6.2.6)$$
$$f(\mathbf{x}) = 0.01x_1^2 + x_2^2$$

subject to,

$$g_1(\mathbf{x}) = 25 - x_1 x_2 \le 0 \qquad (6.2.7)$$
$$g_2(\mathbf{x}) = 25 - x_1^2 - x_2^2 \le 0$$
$$2 \le x_1 \le 50 \qquad (6.2.8)$$
$$0 \le x_2 \le 50$$

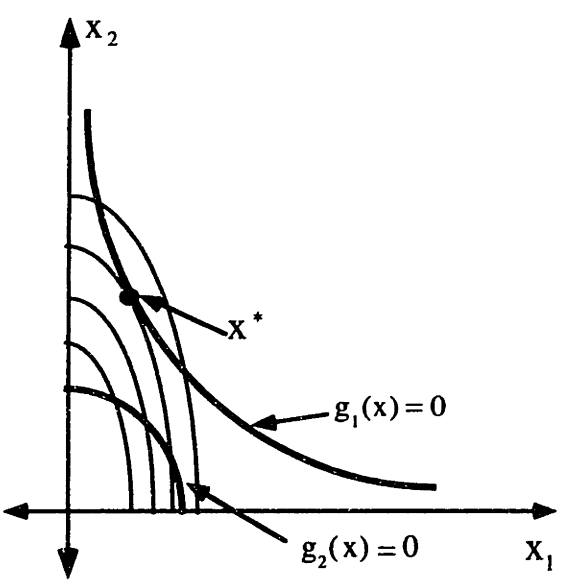


Figure 6.2. Design space and constraints (Example 6.2.3)

Figure 6.2 illustrates the design space and the two constraints. Clearly $g_2(\mathbf{x}) \le 0$ is a redundant constraint. The value of the variables and the objective function at each iteration are tabulated in table 6.3.

Table 6.3. Nonlinear inequality constraints (Example 6.2.3)

Iteration	f(x)	x_1	x_2	$z^t z = c^t c$
1.	101.21	11.0	10.0	-
2.	6.2574	11.039	2.2447	400.00
3.	4.1904	15.854	1.2949	20.129
4.	4.9482	13.817	1.7433	6.660
5.	4.8732	16.919	1.4179	11.760
6.	4.9951	16.151	1.5449	6.456
7.	4.9969	15.761	1.5852	3.031
8.	4.9999	15.825	1.5797	1.0e-4
19.	5.0000	15.811	1.5811	1.0e-06

6.3. Shape and topology synthesis of structures

To illustrate the application of the shape and topology optimization algorithms described in section 5.2.2, we consider the design of planar structural components subjected to two-dimensional loading and boundary conditions. As explained in section 5.2 the feasible region is taken as the initial guess for the shape of the structure so that the shape density function is initialized to the maximum permissible value (unity) all over the feasible region. In the examples considered below we are interested in designing structures made of homogeneous, isotropic materials. In most engineering applications it is difficult to manufacture structures whose density can be varied in a controlled fashion. Therefore, we seek solutions that are fully dense in the sense that the shape density function has the maximum permissible value inside the solid and its value transitions sharply at the boundaries to the lowest possible, that is the threshold value.

Example 6.3.1

In this simple example, we apply forces on a 2 dimensional block that produces bending moment in it. The initial geometry, the applied loading and boundary conditions are illustrated in figure 6.3.

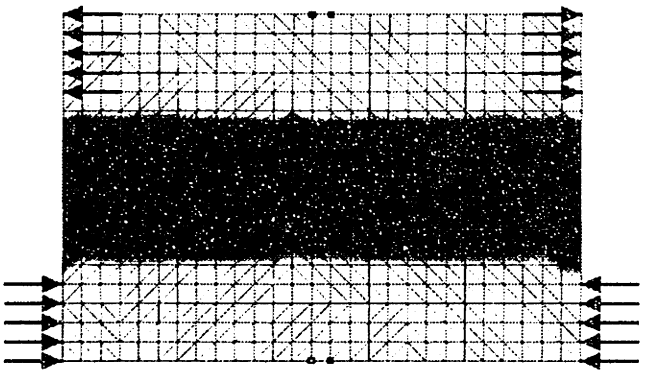


Figure 6.3. Optimal geometry (Example 6.3.1)

The optimal shape is displayed using the contours of the density function. If one interprets these contours as thickness distribution, the optimal shape suggested by the algorithm has an I-shaped cross-section. The result shown in figure 6.3 was obtained using the linear approximate relations described in section 3.4.1. Identical results where obtained using quadratic and higher order relations described in section 3.4.2. In this example, regardless of the material property density relations used, the final result was found to be independent of the quality and density of the mesh used for shape representation and analysis. This may be due to the simple loading applied and the resultant simplicity of the final optimal shape.

Example 6.3.2

We consider here a commonly used test example of designing a two bar frame structure. The solution to this example can by obtained analytically by using simple frame models. For the planar loading case and boundary conditions illustrated in figure 6.4. the optimal geometry has a height twice its width (H=2L).

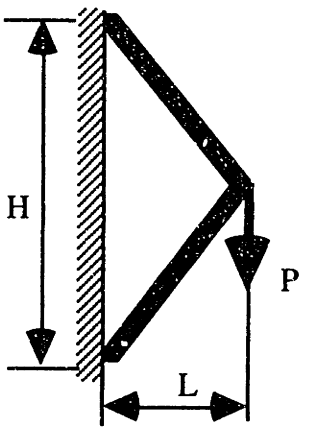


Figure 6.4. Two bar frame structure

In figure 6.5 the optimal shapes computed using the linear material property density relation are illustrated. The shapes shown are the optimal solution computed by removing material in steps of 5% each. The rectangular region represents the initial feasible region. This shape is taken as the starting geometry of a component that supports the shear load shown by the arrow. The highlighted nodes on the boundary are fixed nodes. They represent regions where the structure is supported. In the first step, 5% of the weight is removed by optimizing the shape subject to such a constraint on weight. The dark region has full density ϕ=1 while the lightly shaded region represents an area from which material has been removed. The lightly shaded regions have shape density value below the threshold value of 0.9. At the end of the step, geometry is updated by removing the elements in the lightly shaded area. In the next two steps weight is again reduced by 5% each followed by removal of elements to update the geometry.

As noted in chapter 5, material has to be removed in small steps when the linear material property - density relation (equation 3.4.3a) is used because the threshold value of density has to be set close to 1.0 to prevent material properties from becoming negative. Setting the threshold value close to unity has the additional advantage that the final design is nearly fully dense so that the density values transition sharply at the boundaries from the highest value 1.0, to the lowest (threshold value). However, when large percentage of material removal is desired, many steps of material removal are required. Furthermore, extra computation is required at each step to identify elements where the density is below the threshold and to remove them.

In the example in figure 6.5, the height of the initial feasible region is twice that of the length. The mesh used consisted of 800 elements and 449 nodes. The structure is assumed to be made of a isotropic linear elastic material.

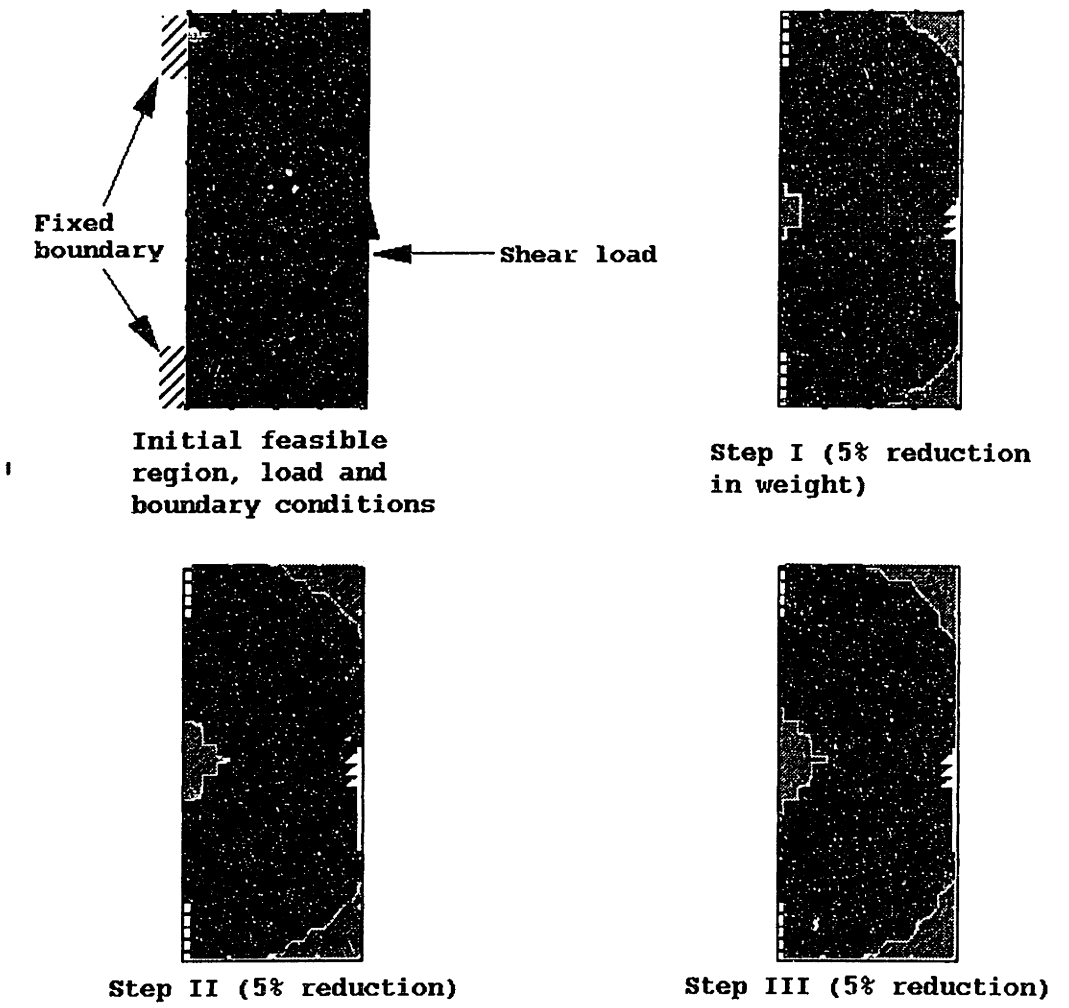


Figure 6.5. Two bar frame (Weight reduction in steps)

Figure 6.6 shows the optimal shape obtained for the same example using a quadratic material property-density relation where the objective function includes penalty on intermediate densities as explained in section 3.4, equation (3.4.6). In this example 60% of the material has been removed from the initial geometry (that is the rectangular feasible domain). Unlike in the linear material-property density relation, material was removed in one step. The shape and topology were first optimized without the penalty on intermediate density values. The optimal geometry thus obtained was taken to be the initial design for a subsequent optimization with penalty imposed on intermediate densities. The optimal design in figure 6.6 was obtained in just two steps, one without penalty and the next with

penalty imposed on intermediate variables. A very regular mesh consisting mostly of equilateral triangles was used in this example. The model consisted of 1081 nodes and 2020 elements. Significant savings in computation is obtained by removing material in one step using quadratic or higher order $d_{ij}(\phi)$ relation, especially when large percentage of material removal is desired. This example took approximately 4 hours of computation on a SGI, Indigo^2 Extreme. Since material was removed in one step very large percentage of material could be removed without incurring large computational expense. Note that even though much smaller percentage of material was removed in the shape obtained in figure 6.5, the optimal topology is essentially the same as in figure 6.6.

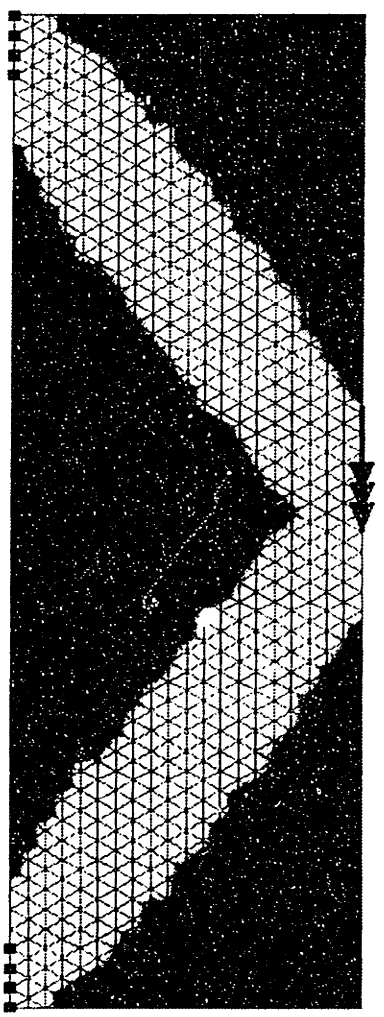


Figure 6.6. Two bar frame (Quadratic $d_{ij}(\phi)$ relation)

Example 6.3.3

Figure 6.7 illustrates another planar structural optimization example where the structure is supported below along the shaded boundary. The loads to be supported are shown by the arrows. The final shape represented by the darkly shaded region was obtained after 4 steps of material removal using the linear approximate $d_{ij}(\phi)$ relation. Note that even though the initial geometry was a rectangular block, the optimization technique automatically identifies regions where it should remove material from and suggests a final shape that is topologically different. A hole was created since the optimization procedure identified this region as a region where material is under-utilized (stresses are lower than in other regions). In this example 1098 elements and 592 nodes were used. Since the loading is symmetric the optimal shapes are expected to be symmetric. For linear $d_{ij}(\phi)$ relation since elements are removed after each step, the quality of mesh affects the solution. For example when the mesh is not symmetric, the elements get removed unsymmetrically resulting in unsymmetric initial geometries for subsequent steps. This biases the solution to become further unsymmetric in subsequent steps. This is a drawback of removing elements at each step. However if perfectly symmetric mesh can be generated for the feasible domain this problem can be overcome.

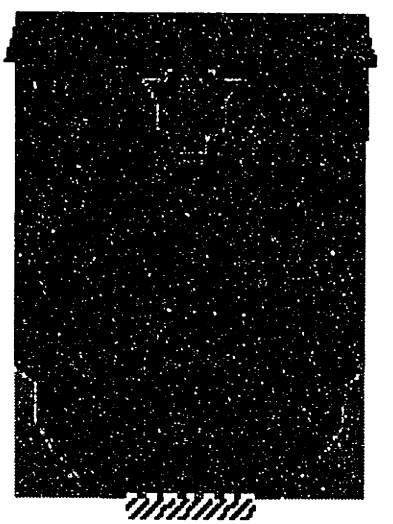


Figure 6.7. Optimal support structure

Quadratic and higher order $d_{ij}(\phi)$ relation is less sensitive to mesh since material is removed in one step. Figure 6.8 shows the same example where a quadratic material property-density relation has been used and 30% of the mass was removed in one step. Note that the optimal topology obtained here is the same as in figure 6.7 even though a different material property density relation was used and a different percentage of material was removed. A regular mesh consisting mostly of equilateral triangles is used in the model which consists of 1131 nodes and 2130 elements. The optimal geometry shown in figure

6.8(a) is not fully dense and the shape density function has intermediate values over large regions near the boundaries. This optimal geometry was used as the starting design for subsequent optimization with a penalty on intermediate densities. This yields the geometry shown in figure 6.8(b) where the shape density function transitions sharply at the boundaries resulting in a fully dense optimal design. The topology suggested by the optimization is the same for both cases in figure 6.8 (a) and (b).

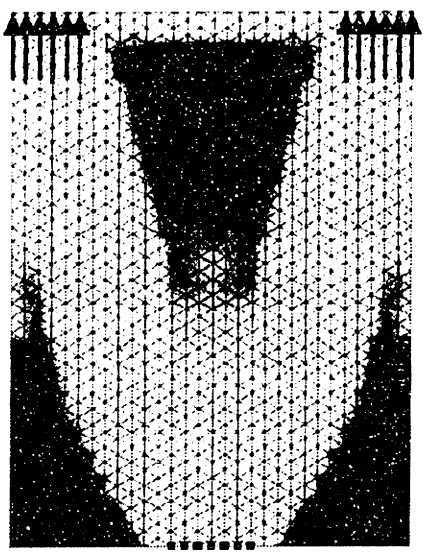


Figure 6.8.(a) No penalty on intermediate densities

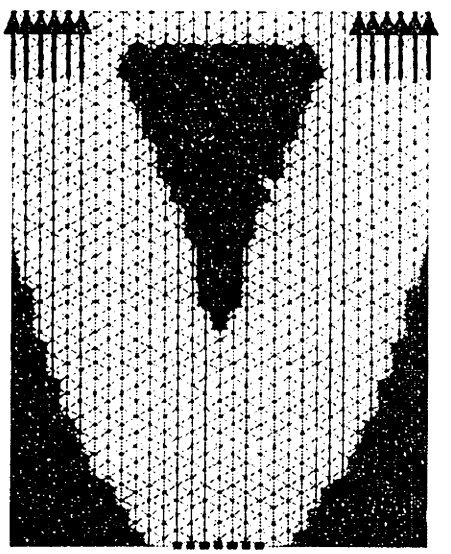


Figure 6.8.(b) With penalty on intermediate densities

Example 6.3.4

In figure 6.9, we consider an example where we are interested in finding the optimal structure that fits in the L-shaped feasible region and supports the load depicted by the arrows. The structure is supported at the shaded boundary. The optimization was carried out assuming the linear material property density relation from equation (3.4.3a). The optimal shape suggested by algorithm after 7 steps of material removal is illustrated by the darkly shaded region in figure 6.9. We used 1950 elements and 1056 nodes. Smoother boundaries can be obtained by using finer mesh.

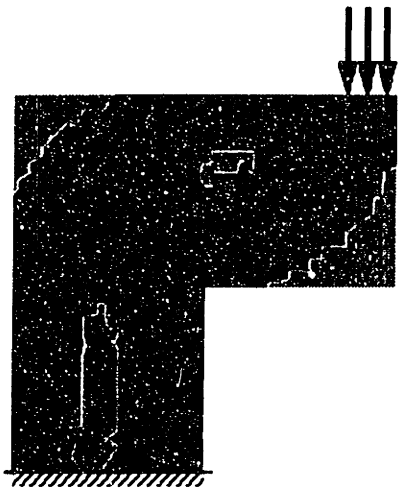


Figure 6.9. Optimal L-shaped support structure

The number of variables again is equal to the number of nodes used since the nodal values of density are treated as design variables. The optimization involved 15-20 iterations in each step of material removal. The optimization algorithm typically performs one function evaluation (finite element analysis) per iteration.

Example 6.3.5

In this example we are interested in designing a frame-like structure for carrying the load indicated in figure 6.10. The feasible load is the rectangular region loaded as in a short cantilever beam. Very large percentage of material has to be removed from the initial geometry to obtain a frame like structures.

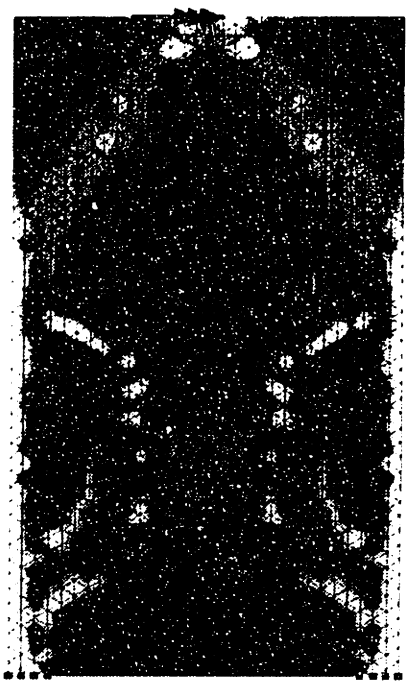


Figure 6.10.(a) 4th order material property density relation

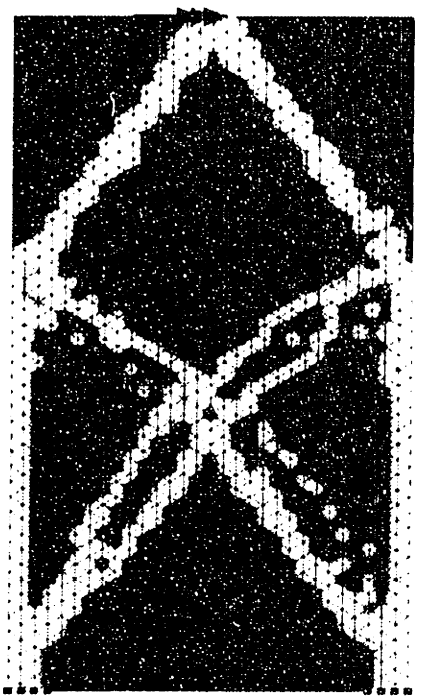


Figure 6.10.(b) With penalty on intermediate densities

The geometries displayed in figure 6.10 were obtained by 70% reduction in mass from the starting shape. The geometry in figure 6.10(a) was obtained using a 4th order material property density relation. Due to the large percentage material removal the final geometry has large regions where the shape density function takes intermediate values. This suggests that the truly optimal shapes for carrying this load may be composite materials whose density varies within the structure. Indeed, naturally occurring optimal structures such as bones in animals have varying density and are infact composite materials. However, for engineering applications structures with varying density are difficult or very expensive to manufacture. To obtain a frame like structure made of fully dense material we apply a penalty on intermediate densities. Figure 6.10(b) shows the structure obtained by adding a penalty to the objective function to eliminate intermediate material. The penalty constant in equation (3.4.6) was increased in 3 steps by a factor of 10 each. The resultant geometry suggests the optimal topology of a frame to support the given load.

The model used in the above example consists of 1441 nodes and 2730 elements. The four steps of optimization required to obtain the result in figure 6.10(b) took roughly 6 hours on an SGI, Indigo workstation. When a large percentage of material needs to be removed such as in this example, linear approximate $d_{ij}(\phi)$ relation is not efficient due to

the large number of material removal steps required. Moreover for this example the results obtained were found to be highly sensitive to the mesh when material was removed in steps followed by removal of elements as in the case of the algorithm described in figure 5.5.

Example 6.3.6

In this example we consider the design of frames to support the loads shown in figure 6.11. Fourth order material property density relation is used in this example followed by three steps of subsequent optimization with penalty on intermediate values of density. The geometry in figure 6.11(a) was obtained after 70% material removal while the geometry in figure 6.11(b) was obtained after 80% material removal from the initial geometry represented by the square feasible region. Note that the topology of the optimal shape is different when the percentage of material removed is different.

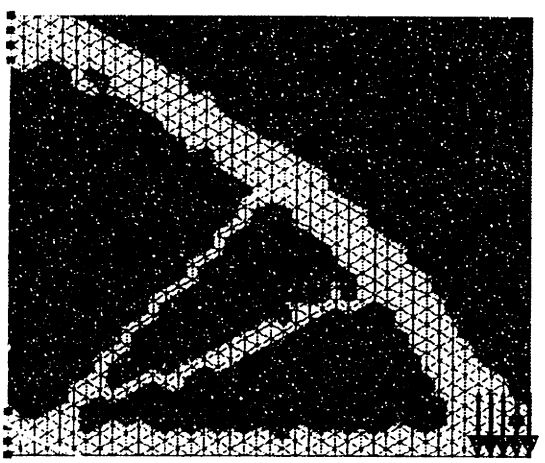


Figure 6.11.(a) 70% material removed

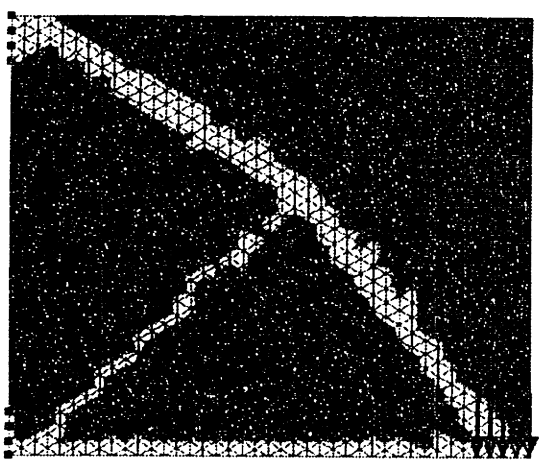


Figure 6.11.(b) 80% material removed

In the above example we have used a regular mesh consisting of 1291 nodes and 2440 elements. The penalty constant was initialized as c_p=10 and was increased by a factor of 10 in the subsequent iterations.

Example 6.3.7

In this example we consider the design of a bridge like frame. The structure is rigidly supported at the two ends and is assumed to carry loads as shown in figure 6.12. The optimal shape was obtained by removing 70% material. A 4th order $d_{ij}(\phi)$ relations was assumed. The optimal shape shown in figure 6.12 was obtained after 6 steps of

optimization with increasing penalty on intermediate densities. To obtain well defined members for the frame a relatively dense mesh consisting of 1597 nodes and 3010 elements was used.

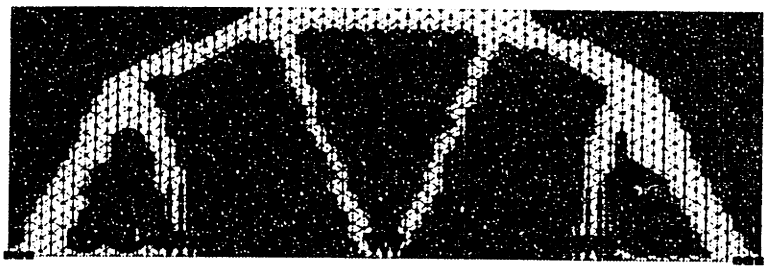


Figure 6.12. Bridge-like frame (70% Material removal)

Example 6.3.8

The example below shows the design of Michell's truss. The structure is rigidly supported at the circular region. The nodes around this region are assumed to be fixed and are therefore highlighted. The circular region may represent a shaft to which the a torque is applied due to the load shown by the arrows. We are interested in designing the optimal structure that transmits the bending moment created by the load. Analytical solution for this problem was given by Michell [Hemp_73]. This example has also been solved using homogenization method by [Suzuki_91]. The model below consists of 1413 nodes and 2616 elements. The optimal shape in figure 6.13 was obtained using fourth order material property density relation and a penalty on intermediate densities. The optimal shape is quite similar to the analytical results obtained by Michell as well as the shapes predicted using homogenization method.

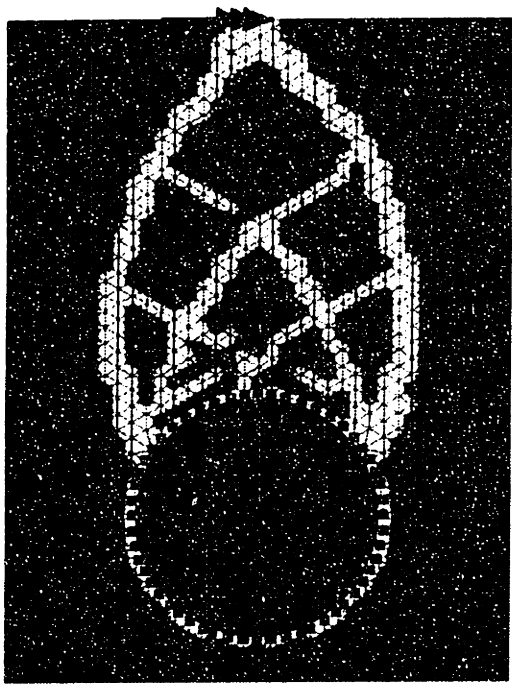


Figure 6.13. Michell's truss (80% Material removal)

Example 6.3.9

In the example below we consider the design of bicycle frame. The loading and boundary conditions are assumed to be as shown in figure 6.14. The structure is supported at the two ends where the wheels are attached and may support load at the seat, handle and the pedal. The design in figure 6.14 was obtained by removing 70% mass from the feasible region. Here again we have used a 4th order material property density relation and applied penalty to intermediate densities in 5 steps.

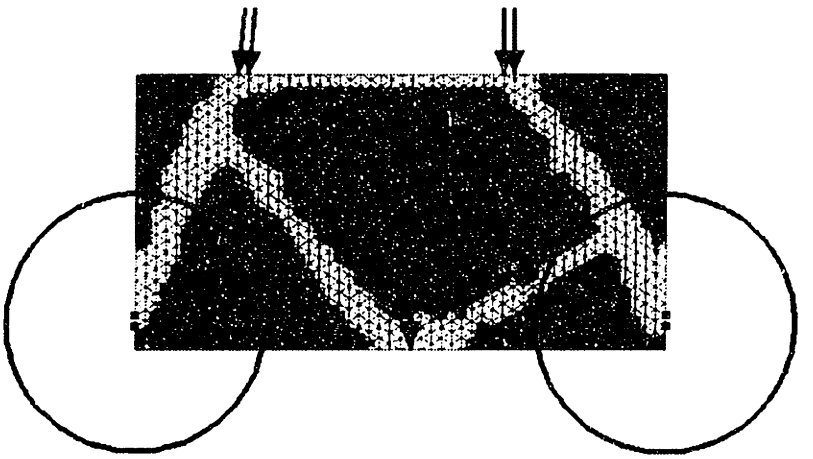


Figure 6.14. Bicycle frame (70% material removal)

6.4. Summary and discussion

In this chapter we have applied the algorithms and concepts discussed in the previous chapters. The MBSLP algorithm was first applied to solve some simple test examples of nonlinear programming. The algorithm was then used in the shape and topology design of structures. The optimal shape and topology are computed by minimizing the compliance of the structure subject to constraint on the total mass of the structure. The constraint is linear and therefore for the examples is section 6.3, we have used the criteria discussed in section 4.3.4 to reset the move limits.

By maximizing the stiffness of the structure subject to constraint on its weight we obtain shapes that better utilize the available material. That is, the optimal shape carries a nearly uniform stress or strain energy distribution all over its domain for the given load. This is clearly illustrated by plotting the strain energy distribution over the domain for the initial shape and the final optimal shape. Figure 6.15 show the strain energy distribution of the two bar frame example (example 6.3.2). The initial shape is the same as the feasible region. It can be seen that the strain energy distribution for this geometry is not uniform. The strain energy is maximum in white regions and least in the dark regions. There are large regions where the strain energy is nearly zero and it is maximum near the applied load and the boundary condition where there is stress concentration. The optimal geometry and the corresponding strain energy distribution is shown on the right. The strain energy is nearly uniform within the optimal shape, except for small regions of stress concentration.

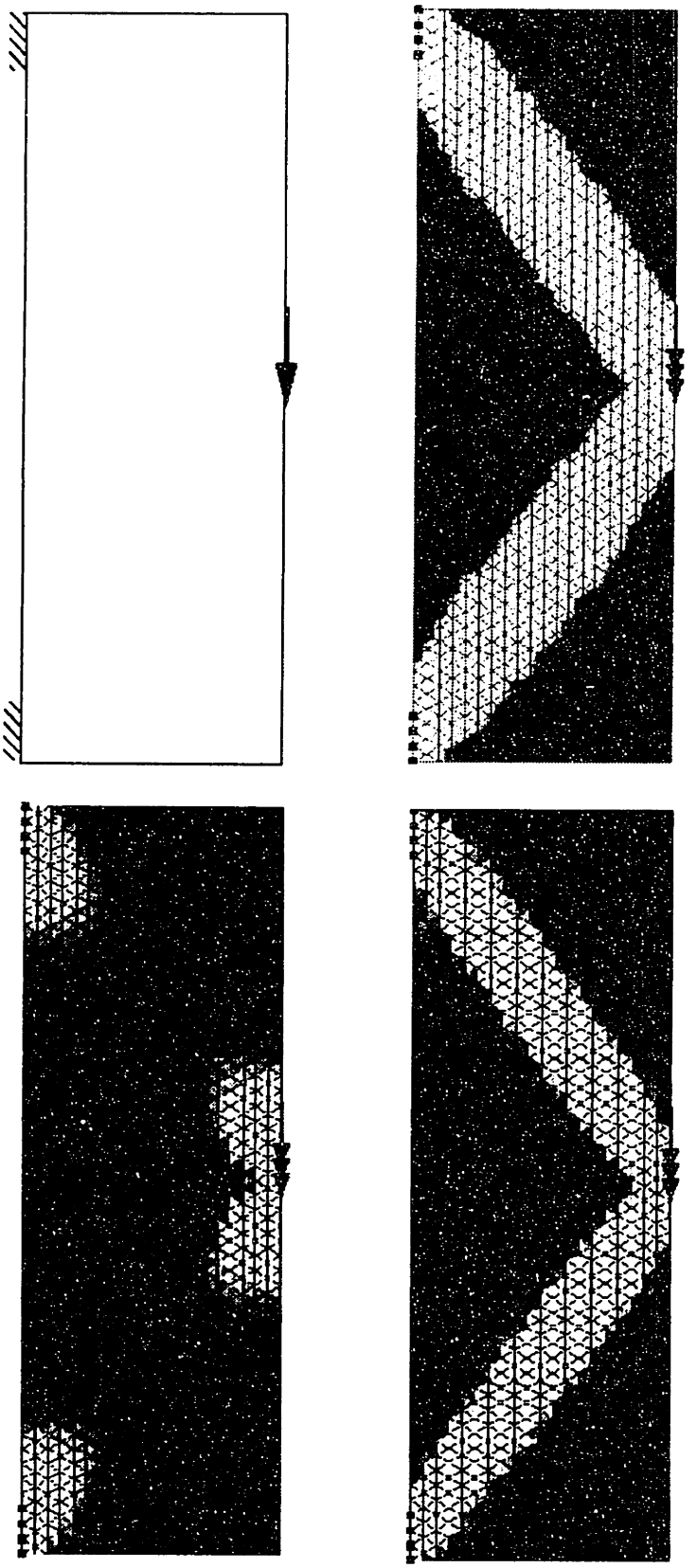


Figure 6.15 Strain energy plot for two-bar frame.

The strain energy distribution is plotted in figure 6.16 for the design of an optimal bracket. The structure is supported below as shown and carries the a load at the circular region. This load is caused due to a shaft passing through the circular hole and is therefore assumed to be a distributed load. The optimal shape obtained by the algorithm is shown on the top right of figure 6.16. Again note that the optimal shape has a nearly uniform strain energy distribution (lower right), while the initial shape on the left has large regions that do not carry significant strain energy. This indicates the available material is more efficiently utilized (fully loaded or stressed) for the optimal topology obtained by minimizing the strain energy absorbed.

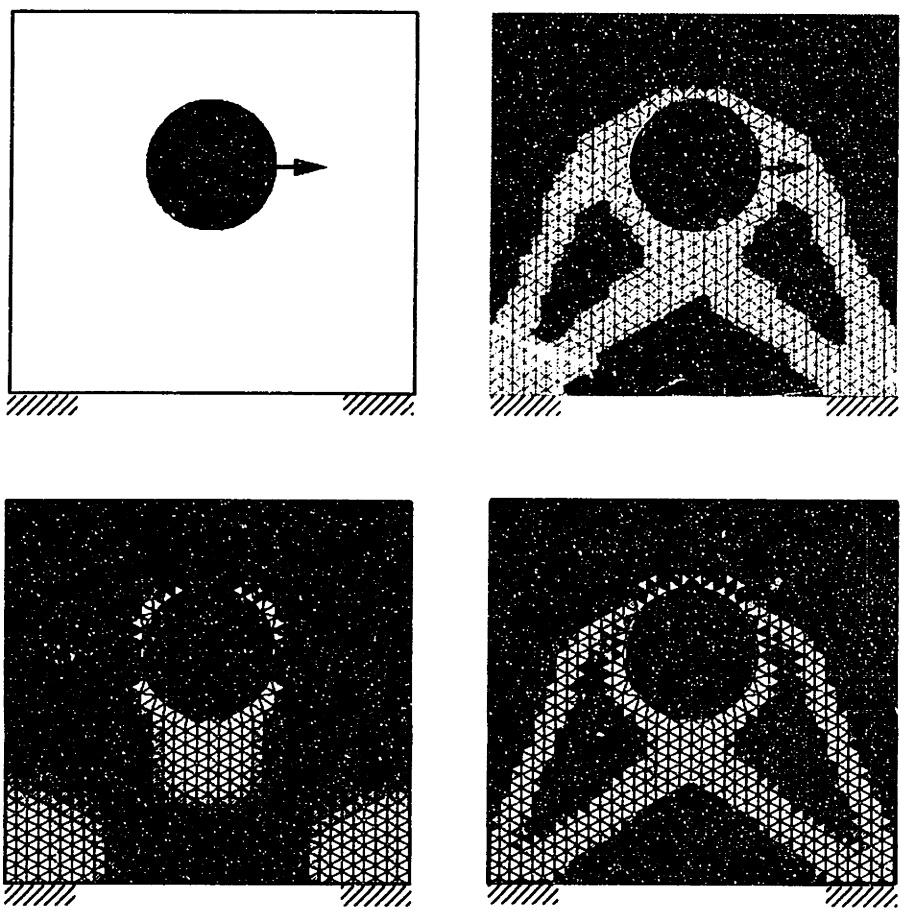


Figure 6.16 Strain energy plots for bracket.

The optimal topology obtained for a given loading and boundary condition is independent of the applied loads. In practice it was found that large magnitude of load led to faster convergence to an optimal solution. When very dense mesh is used the magnitude of the partial derivative of the objective function with respect to the nodal density becomes

very small if the applied loads are not sufficiently large. This seems to lead to loss of accuracy and the resultant poor convergence.

The optimal topology depends on the percentage of material removed from the initial guess, that is the feasible region. When small percentage of material is removed all the material property density relations and corresponding algorithms discussed in chapter 5, lead to the same topology. However, when large percentage of material is removed it is desirable to be able to remove material in one step and therefore quadratic and higher order material property density relation discussed in chapter 3 are more efficient. The resultant optimal shape tend to have large areas of intermediate density suggesting that the truly optimal shapes are composites with varying material property within the structure. To obtain design that are fully dense and homogeneous we add a penalty function to the objective function that penalizes intermediate densities. The optimal shapes obtained by adding the penalty term are frame-like structures that have sharply defined boundaries and fully dense material.

As suggested in chapter 1, shape and topology optimization helps the designer at the conceptual stage of design by computing the optimal shape and topology from purely structural considerations. By maximizing the compliance subject to a constraint on the weight we obtain shapes that are the stiffest possible for the given weight of the structure. This geometry may have to be modified to account for other design considerations. Note that even though the optimal topology is independent of the applied load, the exact shape and the dimensions would depend on the loads if the stress and strain constraints are directly accounted for. However, as noted in chapter 3, the stress and strain constraints are not easy to impose directly since they lead to a large number of constraints and because such constraints make the design sensitive to stress concentration arising out of modeling error. The optimal design obtained by minimizing compliance tend to have nearly uniform stress distribution and the average stress is related to the compliance of the structure. Therefore, it is possible to indirectly ensure that stress constraints are not violated by checking whether the compliance of the optimal structure is above the critical value associated with the design stress.

7 Conclusions

7.1. Thesis Summary

In this thesis the design optimization of structures has been studied with the goal of developing efficient algorithms and methodologies for optimizing both the shape and topology of structural components. In this chapter the main motivations, the key ideas and the results are summarized. The conclusions arrived in the course of this research and a discussion of results obtained are presented in section 7.2. The potential applications and implications of the new trends in structural design is discussed in section 7.3. Future extensions to this research and some general areas that need further research are also discussed in this section.

Structural optimization has traditionally been viewed as the last stage or the detailing stage of design. The conventional techniques of structural optimization treat certain critical dimensions or certain boundaries of the shape as the design variable. This allows very limited change of shape and hence the optimization can often make only very marginal improvements to the initial shape provided by the designer. Combined shape and topology optimization on the other hand treats the entire geometry of the structure as a variable. Rather than make small modifications to an initial design, this approach can predict optimal shapes that are entirely independent of the initial guess. Indeed the optimal shapes predicted by such techniques can be topologically different so that it may have new boundaries and holes.

The introduction of topology optimization fundamentally changes the application and scope of structural optimization. Due to the drastic design changes introduced during combined shape and topology optimization, such techniques can help designers at an early stage of design. Rather than suggest slight modifications to a detailed design, combined shape and topology optimization suggests new design concepts to which details are to be added. This fundamental difference drastically changes the utility and applicability of

structural optimization. By applying structural optimization at an early stage in design, major design modifications are feasible allowing significant improvements in structural performance.

Most structural optimization techniques cannot account for all the design constraints encountered in real life design. Typically only the structural constraints and objectives are modeled so that other constraints such as aesthetics, manufacturing and assembly requirements etc. must be handled manually. Therefore, applying structural optimization techniques at the detailing stage of design can sometimes be counter-productive. Since only minor changes are possible at this stage, often only marginal improvements in structural performance are obtained. Furthermore many design constraints that are not modeled may be violated during the optimization process. Combined shape and topology optimization enables the introduction of structural optimization at the conceptual stage of design. These techniques also account for only the structural design criteria during the optimization process so that the shape obtained is optimal only from structural considerations. However, since the optimization is applied at a early stage, the structurally optimal shape provides a guideline to the designer. The other design criteria still have to be accounted for manually, but in doing so the designer can try to minimize the deviation from the structurally optimal shape.

Shape and topology optimization requires very flexible shape representation that allows the topology to change when the design variables are modified. An implicit shape representation is used in this thesis where the contours of a shape density function corresponding to a threshold value are treated as the boundaries of the shape. The shape density function is defined over a feasible region and is represented by piece-wise linear interpolation over triangular finite elements. The value of the density function at the nodes serve as the design variables of the optimization problem.

The shape as well as the topology can be modified by varying the design variables in the above shape representation. As the shape varies the structural properties should change accordingly. This implies that the material properties should be somehow related to the shape density function. In this thesis many different material property-density relations were studied. In selecting this relation the two major criteria where as follows:

1) When the density decreases in a region the overall stiffness of the structure should decrease.

2) The final designs should be fully dense shapes so that even when the threshold value on the density is close to zero the density of material within the boundary is nearly 1.0.

Shape representations that allow both shape and topology variation typically require large number of design variables. The optimization problem defined in terms of this variables therefore have very high dimension. In addition, structural optimization requires repeated structural analysis. Very efficient algorithms are therefore required to solve these optimization problems. The important requirements for the optimization algorithm are:

1) Its performance should not deteriorate seriously when the number of variables are very large.

2) It should require very few function evaluations.

3) It should exhibit fast convergence.

A sequential optimization algorithm was developed with these requirements in mind. The algorithm can be interpreted as a modified form of sequential linear programming where the move limits are set using the logarithmic barrier method. The key advantage of this method is that the sub problems generated at each iteration is a convex nonlinear program that can be solved using Newton's method. Computation per iteration is further reduced by not solving the sub problem completely. Instead a descent direction of the subproblem is found by Newton's method and the variables are updated using this descent vector. A new subproblem is then defined by re-evaluating the functions for the new value of the variables. The algorithm was found to perform very well for linearly constrained problems. A method for extending the algorithm to handle nonlinear constraints is also proposed in this thesis. However, further research may be required to test its performance and to ensure reliability.

7.2. Conclusions

The implicit shape representation using shape density function along with the moving barrier sequential linear programming algorithm provides a computationally efficient means of combined shape and topology optimization. By selecting approximate material property-density relations we are able to represent the shape density function using piece-wise linear C^0 continuous interpolation. Even though the side constraints on the density variables are set such that they can assume any value in the interval $[\phi_{th},1]$ it is desired that the optimal design be composed of fully dense material since we are interested in designing structures

made of homogeneous, isotropic materials. Therefore, ideally the density value should transition sharply at the boundary from the maximum value (ϕ=1) to the minimum (ϕ=ϕ_{th}).

In this thesis we have experimented with different material property-density relations. A linear approximate relation was first constructed using the assumption that the material properties (Young's modulus and Poisson's ratio) vary linearly with the density function. The material property coefficients in the stress-strain relation were then linearized with respect to density. The threshold value on density was set very close to 1.0 to ensure that the material is fully dense in optimal shapes obtained. Material was removed in small steps so that at the end of each step, the elements in regions where density value is below the threshold are removed from the analysis model to create a hole. This method was found to give good results when only small weight reduction from the initial geometry was required. For large material removal the optimal designs obtained where found to be sensitive to the mesh used. Moreover the method becomes computationally inefficient since material is removed in many small steps and therefore a large number of steps are required to achieve the desired material removal. Another drawback of this material property density relation is that it assumes that the density values decrease monotonically during the optimization process. At the end of each step the elements are removed from regions with low values of density and the density values in these regions are assumed to be fixed. Therefore in subsequent steps the density in such regions cannot increase.

To overcome these difficulties quadratic material property-density relations were used under the assumption that the Young's modulus varies quadratically with respect to density. Similarly higher order relations can be obtained by assuming higher order relation between Young's modulus and the density function. For these relations the material property coefficients do not become negative for density values in the interval [0,1]. Hence the threshold value can be set to zero (or very close to zero). During the optimization process when the density value in a region is reduced to zero the elements in those regions are effectively removed since they do not contribution to the stiffness of the structure. Therefore large percentage of material can be removed in a single step. However, it was found that when large percentage of material is removed the optimal shapes obtained tend to have many regions where the density function has intermediate values. As a results the boundaries are not clearly defined. In order to prevent this a penalty term was introduced in the objective function that penalizes intermediate values of density. The optimization process therefore tends to favor solutions that are fully dense.

In the design of trusses and frames where large percentage of material has to be removed from the initial feasible region, a combination of fourth order material property density relations and a penalty on intermediate density was found to yield good results.

7.3. Future Directions

Shape and topology optimization has the potential to vastly improve the design process of structural components. The traditional methods of shape optimization have so far found application only in aerospace design where weight minimization is a very important design requirement. However, recently optimal design of structural components has become crucial to automobile industry also due to stricter fuel economy standards and stiff competition. Other applications of structural optimization include civil, naval structural design, sports equipment etc. In all these applications topology optimization can potentially reduce design time and yield superior designs by assisting the designer at the conceptual stage of design.

Research in shape and topology optimization is still at an early stage. Possible extensions to the research presented in this thesis are:

1) C^1 continuous interpolation for the shape density function. In combination with higher order material property density relation, such a interpolation function would result in very high order polynomial to be integrated over each element during the assembly of the stiffness matrix.

2) Extension of the methods developed in this thesis to shell elements and 3D elements.

3) Develop better criteria for resetting move limits in the MBSLP algorithm with nonlinear constraints so that the logarithmic barriers better approximate the local curvature of the objective function.

Appendix
Optimality Criteria

In this chapter some simple concepts in optimization theory are described. Details on these concepts can be found in most texts on linear and nonlinear programming such as [Luenberger_84] and [Bertsekas_92].

1. Definitions

Local maxima and minima

The vector $\mathbf{x}^* \in R^n$ is the unconstrained *local minimum* of a function $f(\mathbf{x})$ if the value of the function evaluated at $\mathbf{x}^*$ is less than or equal to the value of the function at all other values of $\mathbf{x}$ in an arbitrarily small neighbourhood of $\mathbf{x}^*$. That is, if there exists a $\varepsilon>0$ such that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \forall\, \mathbf{x} \text{ such that } \|\mathbf{x} - \mathbf{x}^*\| < \varepsilon \tag{A.1}$$

Similarly, a vector $\mathbf{x}^*$ is a unconstrained *local maximum* of the function $f(\mathbf{x})$ if there exists a $\varepsilon>0$ such that

$$f(\mathbf{x}^*) \geq f(\mathbf{x}), \quad \forall\, \mathbf{x} \text{ such that } \|\mathbf{x} - \mathbf{x}^*\| < \varepsilon \tag{A.2}$$

The vector $\mathbf{x}^*$ is the unconstrained *global minimum* of the function $f(\mathbf{x})$, if the value of the function evaluated at $\mathbf{x}^*$ is less that its value for any other $\mathbf{x}$, that is,

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \quad \forall\, \mathbf{x} \in R^n \tag{A.3}$$

These concepts are illustrated in figure A.1 for a one dimensional function. At a strict local minimum the function value is strictly less than at all the neighbouring points.

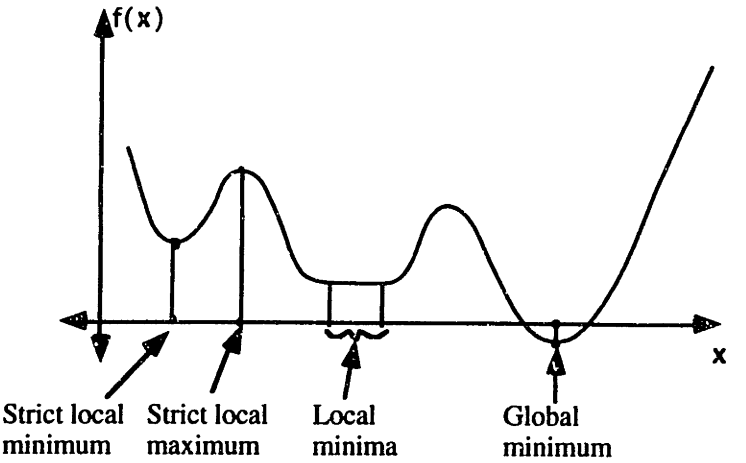


Figure A.1. Unconstrained optima

Descent direction

A vector $\Delta \mathbf{x}$ is said to be a *descent direction* of a function f at the point $\mathbf{x_k}$ if,

$$\nabla f(\mathbf{x}_k)^t \Delta \mathbf{x} < 0 \tag{A.4}$$

Most optimization algorithms find the optimal solution by iterative descent. They start at an initial guess $\mathbf{x}_0$ and then generate a sequence of vectors $\mathbf{x}_k$ such that, $f(\mathbf{x}_{k+1})<f(\mathbf{x}_k)$, k=1,...n. At each iteration a descent direction is computed and the current guess for the optimal solution $\mathbf{x}_k$ is updated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \Delta \mathbf{x} \tag{A.5}$$

where α_k is the step size and $\Delta \mathbf{x}$ is a descent direction. The step size is selected such that

$$f(x_{k+1}) < f(x_k) \tag{A.6}$$

The vector $\Delta \mathbf{x} = -\alpha \nabla f(\mathbf{x}_k)$ is a descent direction of the function at $\mathbf{x}_k$. Similarly, the vector $\Delta \mathbf{x} = -[\mathbf{D}]\nabla f(\mathbf{x}_k)$ is a descent direction if [D] is a positive definite matrix.

Convex sets and functions

Consider a set C which is a subset of R^n. The set C is said to be a *convex set* if

$$\alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in C, \ \forall \ \mathbf{x}_1, \mathbf{x}_2 \in C, \ \forall \ \alpha \in [0,1] \tag{A.7}$$

Therefore, if the set is convex all the points on the line joining any two points in the set will also belong to the set.

Let f be a function defined over a convex set C. The function is said to be a convex function if

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2), \ \forall\, x_1, x_2 \in C, \ \forall\, \alpha \in [0,1] \qquad (A.8)$$

This property of a convex function is illustrated in figure A.2. for a one dimensional case. The linear interpolation $\alpha f(x_1) + (1-\alpha)f(x_2)$ overestimates the value of the convex function at $\alpha x_1 + (1-\alpha)x_2$.

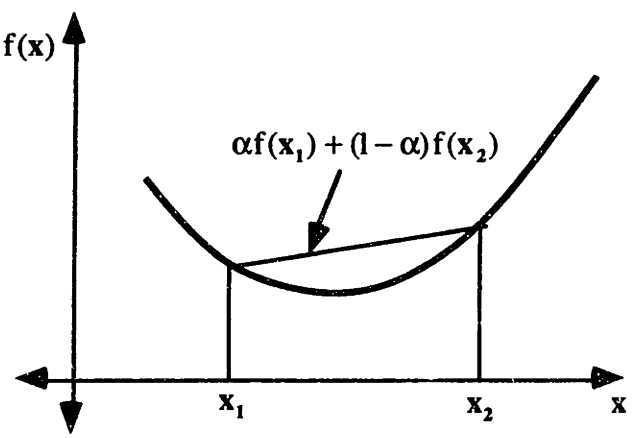


Figure A.2. Property of a convex function

An important property of convex sets is that the intersection of two or more convex sets is also a convex set. Similarly, a weighted sum of two or more convex functions, with positive weights, is convex over the set defined by the intersection of the domains of the functions being added. A linear function is an example of a convex function. A twice differentiable function f defined over a convex set is convex if $\nabla^2 f(x)$ is positive semi-definite for all $x \in C$.

Another property of convex functions that is often used to characterize convexity of differentiable functions is the following. A differentiable function f defined over a convex set C is convex, iff,

$$f(x) \geq f(x_1) + (x - x_1)^t \nabla f(x_1), \ \forall\, x, x_1 \in C \qquad (A.9)$$

If the inequality in equation (A.9) holds strictly whenever $x \neq x_1$, then the function is said to be strictly convex. This property is illustrated in figure A.3.

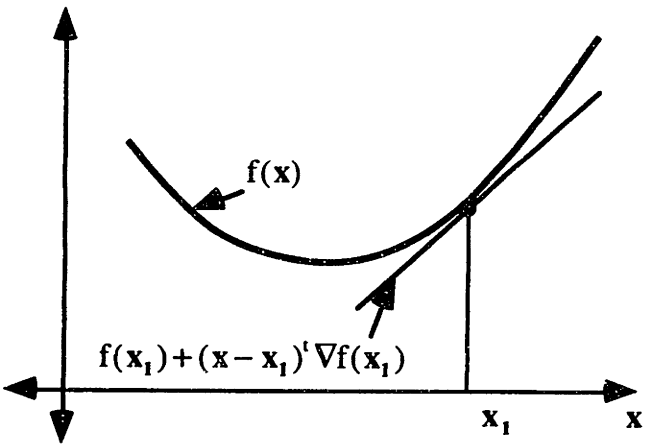


Figure A.3. Property of differentiable convex function

2. Kuhn-Tucker criteria for optimality

The necessary conditions for optimality of a nonlinear program is referred to as the optimality criteria. Consider a general nonlinear program defined below,

$$\min f(\mathbf{x}), \text{ subject to, } \mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{g}(\mathbf{x}) \le \mathbf{0} \tag{A.10}$$

where, $f(\mathbf{x}):R^n \rightarrow R$, $\mathbf{h}(\mathbf{x}):R^n \rightarrow R^m$ and $\mathbf{g}(\mathbf{x}):R^n \rightarrow R^r$ are all nonlinear functions. The optimality criteria for this nonlinear program may be expressed in terms of the Lagrangian function defined as

$$L(\mathbf{x},\lambda,\mu) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{r} \mu_j g_j(\mathbf{x}) \tag{A.11}$$

where $\lambda \in R^m$ and $\mu \in R^r$.

The optimality criteria for the nonlinear program defined in equation (A.10), is also known as the Kuhn-Tucker necessary conditions and can be state as follows:

Let $\mathbf{x}^*$ be a local minimum of the nonlinear program defined in (A.10) and let us assume that the gradients of the constraint functions are linearly independent at this point. Then there exists unique Lagrange multiplier vectors λ^* and μ^* such that,

$$\nabla_x L(\mathbf{x}^*,\lambda^*,\mu^*) = \mathbf{0} \tag{A.12}$$

$$\mu_j^* \ge 0,\ j = 1,\ldots r \tag{A.13}$$

and $\mu_j^* = 0$ for j corresponding to inequality constraints that are active at $\mathbf{x}^*$.

3. Dual problems

Consider the nonlinear program defined as

$$\min\ f(\mathbf{x}),\ \text{subject to},\ \mathbf{g}(\mathbf{x}) \le \mathbf{0},\ \mathbf{x} \in X \tag{A.14}$$

If we refer to this problem as the *primal problem*, its *dual problem* is defined as

$$\max\ q(\mu) \tag{A.15}$$

subject to,

$$\mu_j \ge 0,\ j = 1,..r \tag{A.16}$$

where, $\mu \in R^r$ and q is defined as

$$q(\mu) = \inf_{x \in X} L(\mathbf{x}, \mu) \tag{A.17}$$

$L(\mathbf{x},\mu)$ is the Lagrangian function defined as

$$L(\mathbf{x},\mu) = f(\mathbf{x}) + \sum_{j=1}^{r} \mu_j g_j(\mathbf{x}) \tag{A.18}$$

and the function $q(\mu)$ is defined over the domain $\mu \in D_q = \{\mu |\ q(\mu) > -\infty\}$.

References

[Allaire_92] G. Allaire and R. V. Kohn, "Topology optimization and optimal shape design using homogenization", *Topology Design of Structures*, Ed. Bendsoe and Mota Soares, Kluwer Academic Publishers, 1992.

[Anagnostou_92] G. Anagnostou, E.M. Ronquist and A.T.Patera, "A Computational Procedure for Part Design", *Computer Methods in Applied Mechanics and Engineering.*, vol. 97, pp. 33-48, (1992).

[Bathe_82] K. J. Bathe, *Finite Element Procedures in Engineering Analysis*, Civil engineering and engineering mechanics series, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[Bennett_86] J. A. Bennett and M. E. Botkin, Editors, "The Optimal Shape: Automated Structural Design", General Motors Research Laboratories Symposia Series, Plenum Press, New York, 1986.

[Bendsoe_92a] Martin P. Bendsoe and Carlos A. Mota Soares, Editors, *Topology Design of Structures*, Kluwer Academic Publishers, 1992.

[Bendsoe_92b] M. P. Bendsoe, A. Diaz and N. Kikuchi, "Topology and generalized layout optimization of elastic structures", *Topology Design of Structures*, Ed. Bendsoe and Mota Soares, Kluwer Academic Publishers, 1992.

[Bendsoe_91] M.P. Bendsoe and H.C. Rodrigues, "Integrated Topology and Boundary Shape Optimization of 2-D solids", *Computer Methods in Applied Mechanics and Engineering.*, 87 (1991) 15-34.

[Bendsoe_88] M.P. Bendsoe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method", *Computer Methods in Applied Mechanics and Engineering*, Vol. 71, pp. 197-224, 1988.

[Bensoussan_78] A. Bensoussan, J. Lions and G. Papanicolaou, *Asymptotic analysis for periodic structures*, North-Holland Publishing Company, New York, 1978.

[Bertsekas_92] Dimitri P. Bertsekas, "Notes on Nonlinear Programming", MIT course 6.252 notes, 1992.

[Botkin_86] M. E. Botkin, R. J. Yang and J. A. Bennett, "Shape optimization of three-dimensional stamped and solid automotive components", *The Optimum Shape*, J. A. Bennett and M. E. Botkin Editors, Plenum Press, NY, 1986.

[Briabant_84] V. Briabant and C.Fleury, "Shape optimal design using B-splines", *Computer Methods in Applied Mechanics and Engineering*, 44 (3) (1984) 247-267.

[Choi_87] K. K. Choi, "Shape Design Sensitivity Analysis and Optimal Design of Structural Systems", in *Computer Aided Optimal Design: Structures and Mechanical Systems* (C.A. Mota Soares, Ed.), Springer Verlag, pp. 439-492, (1987).

[Fluery_89] C. Fleury, "Efficient approximation concepts using second order information", *International Journal of Numerical Methods in Engineering*, Vol. 28, pp. 2041-2058, 1989.

[Fluery_86] C. Fluery and V. Briabant, "Structural Optimization: A new dual method using mixed variables", *International Journal of Numerical Methods in Engineering*, Vol. 23, pp. 409-428, 1986.

[Fluery_79] C. Fleury, "Structural weight optimization by dual methods of convex programming", *International Journal of Numerical Methods in Engineering*, Vol. 14, pp. 1761-1783, 1979.

[Fukushima_92] J. Fukushima, K. Suzuki and N. Kikuchi, "Topology optimization of a Car Body with Multiple Loading Conditions, The 33rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 2499/2507, (1992).

[Gibbs_76] N. E. Gibbs, W. G. Poole,Jr., and P. K. Stockmeyer, "An algorithm for reducing the bandwidth and profile of a sparse matrix", SIAM, Journal of Numerical Analysis, Vol. 13, No. 2, 1976.

[Haug_86] E. J. Haug, K. K. Choi, V. Komkov, "Design Sensitivity Analysis of Structural Systems", Mathematics in Science and Engineering series, Academic Press Inc., 1986.

[Haftka_92] R.T. Haftka and Z. Gurdal, *Elements of Structural Optimization*, 3rd Edition, Kluwer academic publishers, 1992.

[Haftka_86] R.T. Haftka and R.V. Grandhi, "Structural shape optimization - A survey", *Computer Methods in Applied Mechanics and Engineering.*, vol. 57, pp. 91-106, (1986).

[Hemp_73] W. S. Hemp, *Michells structural continua*, Optimal structures, Clarendon Press, Oxford, 1973.

[Hernandez_91] S. Hernandez and C.A. Brebbia, *Optimization of Structural Systems and Industrial Applications*, Computer Aided Optimum Design of Structures 91, Computational Mechanics Publications, Elsevier Applied Science, 1991.

[Hock_81] Willi. Hock, and Klaus Schittkowski, *Test examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, New York, 1981.

[Ho-Le_88] K. Ho-Le, "Finite element mesh generation methods: a review and classification", *Computer-Aided Design*, Vol. 20, no. 1, pp. 27-38.

[Imam_82] M. H. Imam, "Three-dimensional shape optimization", *International Journal of Numerical Methods in Engineering*, Vol. 18, pp. 661-673, 1982.

[Kirsch_81] Uri Kirsch, "Optimum Structural Design: Concepts, Methods and Applications", McGraw-Hill Book Company, NewYork, 1981.

[Kikuchi_86] N. Kikuchi, *Finite element methods in mechanics*, Cambridge University Press, New York, 1986.

[Kohn_86] R.V. Kohn and G. Strang, "Optimal design and relaxation of variational problems", *Communications in Pure and Applied Mathematics*, vol. 39, pp. 113-137 (Part I), pp. 139-182 (Part II), and pp. 333-350 (Part III), (1986).

[Kumar_93] Ashok V. Kumar and David C. Gossard, "A sequential approximation technique for nonlinear programming using logarithmic barriers" to be submitted to *International Journal of Numerical Methods in Engineering*, 1993.

[Kumar_92] Ashok V. Kumar and David C. Gossard, "Geometric modeling for shape and topology optimization", Fourth IFIP WG 5.2, *Geometric modeling in Computer-Aided Design.*, Ed. Wozny, M. et. al. 1992.

[Lawson_77] C. L. Lawson, "Software for c1 interpolation", in Rice J. (ed.), *Mathematical Software III*, Academic Press, New York, 1977.

[Luenberger_84] Luenberger, "Introduction to Linear and Nonlinear Programming", Addison-Wesley, 1984.

[Monteiro_89a] Monteiro, R.D.C. and Adler, I., "Interior path following primal-dual algorithms. Part I: Linear Programming", *Mathematical Programming*, Vol. 44, pp. 27-41, 1989.

[Monteiro_89b] Monteiro, R.D.C. and Adler, I., "Interior path following primal-dual algorithms. Part II: Convex Quadratic Programming", *Mathematical Programming*, Vol. 44, pp. 27-41, 1989.

[Morris_82] A. J. Morris (Editor), "Foundations of Structural optimization: A Unified Approach", John Wiley and Sons, 1982

[Mota Soares_87] Carlos A. Mota Soares, Editor,*Computer Aided Optimal Design: Structures and Mechanical Systems* , NATO ASI Series, Springer Verlag, Series F: Computer and Systems Science, Vol. 27, (1987).

[Papalambros_90] Panos Papalambros and Mehran Chirehdast, "An Integrated Environment for Structural Configuration Design", *Journal of Engineering Design*, Vol. 1(1): pp. 73-96, 1990.

[Press_88] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, "Numercal Recipes in C, The art of Scientific Computing", Cambridge Universty Press, NY, 1988.

[Reddy_84] J. N. Reddy, *An introduction to the finite element method*, McGraw-Hill, New York, 1984.

[Rozvany_91] G. I. N. Rozvany, Editor, Optimization of Large Structural Systems, Volume I-II, NATO ASI Series, Kluwer Academic Publishers, 1991.

[Schmit_74] L.A. Schmit and B. Farshi, "Some Approximation concepts for Structural Synthesis", AIAA Journal, Vol. 12(5), pp. 692-699, 1974.

[Shigley_83] Joseph E. Shigley and Larry D. Mitchell, "Mechanical Engineering Design", Fourth Edition, Series in Mechanical Engineering, McGraw-Hill Book Company, New York, 1983.

[Sloan_87] S. W. Sloan, "A fast algorithm for constructing Delaunay triangulations in the plane", *Advanced Engineering Software*, Vol. 9, No. 1, 1987.

[Strang_86a] Gilbert Strang and R.V. Kohn, "Optimal design in elasticity and plasticity", *International Journal for Numerical Methods in Engineering,*, vol. 22, pp. 183-188, (1986).

[Strang_86b] Gilbert Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Cambridge, Massachusetts, 1986.

[Suzuki_91] K. Suzuki. and N. Kikuchi, "A homogenization method for shape and topology optimization", *Computer Methods in Applied Mechanics and Engineering*, Vol. 93, pp. 291-318, 1991.

[Svanberg_87] K. Svanberg, "The Method of Moving Asymptotes - A new method for structural optimization", *International Journal of Numerical Methods in Engineering*, Vol. 24, pp. 359-373, 1987.

[Topping_83] B. H. V. Topping, "Shape optimization of skeletal structures: A review", *Journal of Structural Engineering*, Vol. 109, No. 8, pp. 1933-1951, 1983.

[Yang_86] R. J. Yang, K. K. Choi and E. J. Haug, "Numerical considerations in structural component shape optimization", ASME Journal of Mechanics, Transmissions and Automation in Design, Vol. 107, No. 3, pp. 334-339, 1986.

[Zienkiewicz_89] O. C. Zienkiewicz, *The finite element method*, 4th edition, McGraw-Hill, NY, 1989.