ENGINEERING OPTIMIZATION

A projekt keretében elkészült tananyagok:

Anyagtechnológiák

Materials technology

Anyagtudomány

Áramlástechnikai gépek

CAD tankönyv

CAD book

CAD/CAM/CAE elektronikus példatár

CAM tankönyv

Méréstechnika

Mérnöki optimalizáció

Engineering optimization

Végeselem-analízis

Finite Element Method

Budapest University of Technology and Economics
Faculty of Mechanical Engineering

Óbuda University
Donát Bánki Faculty of Mechanical and Safety Engineering

Szent István University
Faculty of Mechanical Engineering

ENGINEERING OPTIMIZATION

Course bulletin

Editor:
GÁBOR KÖRTÉLYESI

Authors:
CSILLA ERDŐSNÉ SÉLLEY
GYÖRGY GYURECZ
JÓZSEF JANIK
GÁBOR KÖRTÉLYESI

TYPOTEX

2012

READERS: Prof. Dr. Károly Jármai

ISBN 978-963-279-686-4
PREPARED UNDER THE EDITORSHIP OF Typotex Publishing House
RESPONSIBLE MANAGER: Zsuzsa Votisky

GRANT:

Made within the framework of the project Nr. TÁMOP-4.1.2-08/2/A/KMR-2009-0029, entitled „KMR Gépészmérnöki Karok informatikai hátterű anyagai és tartalmi kidolgozásai" (KMR information science materials and content elaborations of Faculties of Mechanical Engineering).

KEYWORDS:

shape optimization, topology optimization, sensitivity analysis, Design of Experiments, engineering optimization, robust design, optimization of production processes, constrained optimization, gradient methods, multiobjective optimization, evolutionary optimization methods

SUMMARY:

The handout presents the design optimization tasks occurring in the mechanical engineering practice and clarifies the basic concepts needed to set up optimization model.
The optimization algorithms are explained from simple univariate and unconditional techniques up to the multivariate and conditional methods based on these, taking into account widely usable direct methods, gradient methods which have favorable convergence properties and random procedures having chance to find the global optimum. Encountering with complex engineering optimization problems in the sensitivity analysis helps to reduce the size of the task, but in given time to solve them the design of experiments methods are the natural choice. The uncertainties in engineering systems can be treated by the robust design. The available optimization softwares are shown as practical implementation of these methods and techniques to support the mechanical engineering design. After the methods assisting the planning process (optimization machine parts and components), we are looking at the optimization possibilities of production processes. Applying a complex systems-based model on enterprise level, we learn how to be utilized high-quality machines in the production process and objective decision-making, supplying information to complex technical-economic qualification.

CONTENTS

1. Introduction 9
1.1. The structure of the book 9
1.2. Optimization in the design process 10
1.3. The basic elements of an optimization model 11
1.3.1. Design variables and design parameters 11
1.3.2. Optimization constraints 12
1.3.3. The objective function 14
1.3.4. Formulating the optimization problem 15
1.4. Optimization examples 16
1.5. References 18
1.6. Questions 18

2. Single-variable optimization 19
2.1. Optimality Criteria 19
2.2. Bracketing Algorithms 19
2.2.1. Exhaustive Search 19
2.2.2. Bounding Phase Method 20
2.3. Region-Elimination Methods 20
2.3.1. Golden Section Search 20
2.4. Methods Requiring Derivatives 22
2.4.1. Newton-Raphson Method 22
2.4.2. Bisection Method 23
2.4.3. Secant Method 24
2.5. References 24
2.6. Questions 24

3. Multi-variable optimization 25
3.1. Optimality Criteria 25
3.2. Direct Search Methods 25
3.2.1. Simplex Search Method 25
3.2.2. Powell's Conjugate Direction Method 26
3.3. Gradient-based Methods 29
3.3.1. Numerical Gradient Approximation 30
3.3.2. Cauchy's Method (Steepest Descent) 30
3.3.3. Newton's Method 31
3.3.4. Marquardt's Method 31
3.3.5. Variable-Metric Method (Davidon-Fletcher-Powell method) 31
3.4. References 34
3.5. Questions 35

4. Constrained optimization 36
4.1. Kuhn-Tucker Conditions 36
4.2. Transformation Methods 37
4.2.1. Penalty Function Method 37
4.2.2. Method of Multipliers 42

4.3. Constrained Direct Search 43
4.3.1. Random Search Methods 43
4.4. Method of Feasible Directions 43
4.5. Quadratic Approximation Method 50
4.6. References 51
4.7. Questions 51

5. Nontraditional optimization techniques 52
5.1. Genetic Algorithms 52
5.1.1. Fundamental Differences with Traditional methods 52
5.1.2. Reproduction Operator 52
5.1.3. Crossover Operator 53
5.1.4. Mutation Operator 54
5.2. References 61
5.3. Questions 61

6. Multi-criterion optimization 62
6.1. Multi-Objective Optimization Problem 62
6.2. Principles of Multi-Objective Optimization 63
6.3. Illustrating Pareto-Optimal Solutions 65
6.4. Objectives in Multi-Objective Optimization 69
6.5. Non-Conflicting Objectives 69
6.6. References 70
6.7. Questions 70

7. Optimization of products and machine components 71
7.1. The place and role of the optimization in the design process 71
7.2. The main types of optimization tasks, incorporating them into the design process 72
7.3. The optimization examples in the fields of mechanical engineering 75
7.4. Questions 77

8. Grouping and evaluation of the methods for solving engineering optimization tasks 78
8.1. Optimization tasks containing only geometric conditions 78
8.2. Solving optimization tasks using heuristic optimization procedure 78
8.3. Optimality Criteria (OC) method for solving stress concentration problems 80
8.4. Mathematical Programming Methods (MP) 83
8.4.1. Lagrange function and the Kuhn-Tucker's criteria 83
8.5. Comparing the different methods 84
8.6. References 84
8.7. Questions 84

9. The role and method of sensitivity analysis; case study 86
9.1. The direct and adjoint techniques of sensitivity analysis 87
9.1.1. Direct sensitivity calculation 87
9.1.2. Adjoint method 87
9.2. Sensitivity analysis of crank arm– main steps 88
9.3. Questions 95

10. Shape optimization. Geometric parameters and their impact on the optimum 96
10.1. The effect of the geometry description on the optimization model - a case study 96
10.2. Questions: 99

11. Topology optimization 100
11.1. Place of topology optimization in the design process 100
11.2. Benchmark problems for testing the algorithms 101
11.3. Methods of topology optimization 102
11.4. Homogenization method 103
11.5. The SIMP method 103
11.6. Level set methods(LSM) 108
11.7. Evolutionary Structural Optimization (ESO) 110
11.8. Nonprofit software tools to obtain the optimal topology 110
11.8.1. TOPOPT 111
11.8.2. TOPOSTRUCT 112
11.9. Summary 113
11.10. Literature 114
11.11. Questions 115

12. Optimization methods of engineering problems 116
12.1. Integration of optimization into the design process 116
12.2. Delimiting the solution time – meta-models, design of experiments 117
12.2.1. Data mining and sensitivity analysis for reducing the problem size 118
12.2.2. Design of Experiment 118
12.2.3. Full factorial design 120
12.2.4. Central Composite Design 121
12.2.5. Random and Latin hypercube design 122
12.2.6. Box-Behnken design 123
12.3. Metamodels 124
12.3.1. Response surface method 124
12.3.2. Kriging metamodel 125
12.3.3. Metamodel with radial basis function 126
12.4. Neural networks 127
12.5. Global optimum 128
12.6. Multidisciplinary optimization 129
12.7. Robust design 129
12.8. Dynamic problems 134
12.9. Acceleration of computation 134
12.10. Summary 134
12.11. Literature 135
12.12. Questions 135

13. Software tools for structural optimization 136
13.1. Design cycle, possibility of modelling 136
13.2. Structure of software, demands 137
13.3. Commercial software tools in the design optimization 138
13.4. Summary 139
13.5. Questions 139

14. Optimizing production processes 140
14.1. Introduction 140
14.1.1. Questions, tasks: 144
14.2. Optimization, optimum, suboptimum 144
14.2.1. Questions, tasks: 147

14.3. Intersection as suboptimum 147
14.3.1. Questions, tasks. 154
14.4. The idea of the system, its interpretation 154
14.4.1. Questions, tasks. 157
14.5. The system theoretic model of the company 157
14.5.1. Interaction of the machine maintenance and company profit 162
14.5.2. Practical experiences of the model's application in companies 165
14.5.3. Summary 171
14.5.4. Questions, tasks 172
14.6. Organizing mechanical engineering processes 173
14.6.1. Organizational forms, models of mechanical engineering 176
14.6.2. General model to plan a product to the production process. 182
14.6.3. Cost efficient optimizing methods, models 188
14.6.4. Questions, tasks. 210
14.7. Complementary examples to the subject 211
14.8. Literature: 223

1. INTRODUCTION

This electronic book was supported by TÁMOP-4.1.2/08/2/A/KMR. Basically this curriculum is addressed to the B.Sc. students but the authors hope that the other readers will interested in this book.
To understand the main fields of this ebook, some mathematical, basic informatics, mechanical, CAD and finite element background knowledge is required. The necessary CAD and finite element knowledge can be found in two other electronic books, which are also developed in the frame of the TÁMOP-4.1.2/08/2/A/KMR project so we refer to these.
In the libraries and on the internet there are lots of books, educational themes in this topic, which all show that this field is very popular and important. The industrial specialists show interest in the optimization researches so it is important that the results of research should be applied in wide range. Nowadays the optimization software's are not only a research tool, but they are included into the commercial software packages becoming basic part of the design process. The other trend of the development is that every industrial finite element software packet has optimization module.
In the light of this development, the following question arises: what is the difference between this electronic book and the hundreds of optimization books can be found in the libraries? The authors trying to introduce the solution method of real world CAD based optimization problems, arising in the mechanical engineering practice. Overcome to the simple analytical optimization problems we will focus on the different numerical solution techniques which can be advantageously applied during the optimization of the real machine parts.

1.1. The structure of the book

In the introduction section of this electronic book the formulation of the optimization problem will be discussed, clarifying the basic definitions. Then we present some typical optimization problems, showing a wide range of applicability of this theme.

In the first part of the educational material we introduce the most important basic optimization methods. Of course, there is no opportunity to introduce all type of optimization methods in details, so we are going to focus on the procedures used by later on solving the CAE based optimization problems.

In the next chapter, the problem classes (and the suggested solution techniques) will be discussed in the case of solving mechanical engineering problems. The different methods and usage of the sensitivity analysis is also presented, and the role of the topology-and shape optimization in the whole design process will be shown. In this section we are dealing with the analysis of complex practical problems, and briefly describe the capabilities of industrial systems, which can be illustrated with an example.

The optimization tasks are not only arises in the design phase of the products, but they have a great importance in the production phase too. The best designed machine can be unsalable, without an economic production technology. In this chapter, the various processes presented an analysis of the optimal interaction, there are also highlighted the importance of optimum and suboptimum. Complex system-levels model through the use of company will show that

how the high-quality machines can be economically used in the production process. We will present typical manufacturing process planning optimization examples.

1.2. Optimization in the design process

In the design process (from the basic ideas to the detailed digital mockup of the product), many aspects should be taken into account.
These requirements are usually connected to safety or economy of the product, but also the manufacturing, using repairing considerations come to the fore. For example the designer has to develop cheapest product, while a large number of additional conditions should be met.

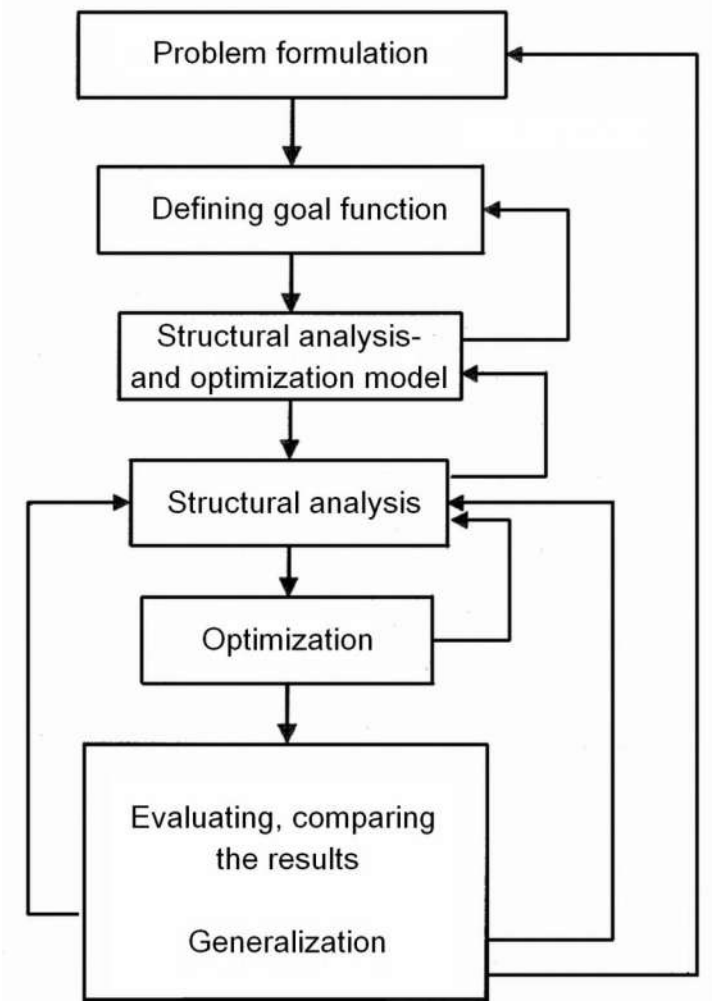

Figure 1.1.: The main steps of solving engineering optimization problem [1.1]

A real design process is usually not a linear sequence, but some steps are repeated, to obtain better solution. The main steps of solving engineering optimization can be seen in Figure 1. It is worth to remark, that the result of the design process, is depending on the accuracy of the introduced steps. For example, if a shape optimization should be taken and the structural responses are calculated numerically, than the error of the structural analysis, can result inaccu-

racy in the optimized shape too. The error can come from mesh density or form the incorrect modeling of the real boundary conditions as well.

Last time the usage of 3D CAD systems and numerical structural analysis techniques has been increased in the industrial applications. This is because; using these tools wide range of design processes can be covered. On the other hand the "time to market" became first objective so the computer-aided technique arises in the early stages of the design process. The modern numerical simulation tools provide an opportunity to decrease the number of the costly and time consuming physical experiments. Although the advanced numerical simulation tools and optimization procedures has a significant role in the product development in the reduction of the time, but the whole design process cannot be automated.
In case of solving structural optimization task we can choose from analytical and numerical methods. The analytical methods are often not suitable for the analysis of complex engineering problems, so this educational material mainly deals with the numerical techniques of the structural optimization. The numerical procedures based on iterative searching techniques which lead to an approximate solution. The searching process continues until the so-called convergence condition is satisfied, so we are sufficiently near the optimal solution.

1.3. The basic elements of an optimization model

1.3.1. Design variables and design parameters

A structural system can be described by a set of quantities, some of which are viewed as variables during the optimization process. Those quantities defining a structural system that are fixed during the automated design are called preassigned parameters or design parameters and they are not varied by the optimization algorithm. Those quantities that are not preassigned are called design variables. The preassigned parameters, together with the design variables, will completely describe a design.

From the mathematical point of view three types of design variables can be distinguished:

The design variables can be considered as continuous or discrete variables. Generally it is easier to work with continuous design variables, but a part of the real world problems contains discrete type of design variables. An intermediate solution, when we know that a large number of discrete design variable values should be considered, then it will be categorized as pseudo discrete. In this case, we solve the task considering this variable a continuous design variable and after the solution the closest possible discrete values will be checked.

From the physical point of view there are four types of the design variables:

1. Mechanical or physical properties of the material (Material design variable)
Material selection presents a special problem. Conventional materials; have discrete properties, as for example a choice is to be made from a discrete set of materials. If there have a few numbers of materials, the task is easier, if we perform the structural analysis for each material separately and comparing the results to choose the optimum material. A typical application is for reinforced composite materials to determine the angles of the reinforcements. Such type of design variables can be considered to be continuous ones.

2. topology of the structure, connecting members of the scheme, or the number of members of the interface schema; (Topology Design Variables)
The topology of the structure can be optimized automatically in certain cases when members are allowed to reach zero size. This permits elimination of some uneconomical members during the optimization process. An example of a topology design variable is if we looking for the optimal truss structure considering one design variable for each truss element (1 if the member exists or 0 if the member is absent). This type of design variables, according to the mathematical classification is not continuous.

3. The shape of the structure (Configurational or Geometric Design Variables)
This type of design variable leads us to the field of shape optimization. In case of machine design application, the geometry of the part should be modified close to the stress concentration areas, in order to reduce the stresses. On other hands, the material can be removed in the low stress areas, in order to make the structure lighter. So we are looking the best possible shape of the machine part. For example the variable surface of the structure can be described by B-Spline surfaces and the control nodes of such splines can be chosen as a design variable. This is a typical example for shape optimization, and these types of design variables are usually belong to the continuous category.

4. Cross-Sectional Design Variables or the dimensions of the built-in elements
Mainly for historical reasons, size-optimization is a separate category, which has got the simplest design variables. For example, the cross-sectional areas of the truss structure, the moment of inertia of a flexural member, or the thickness of a plate are some examples of this class of design variable. In such cases the design variable is permitted to take only one of a discrete set of available values. However, as discrete variables increase the computational time, the cross- sectional design variables are generally assumed to be continuous.

The design variables and design parameters are together clearly define the structure. If the design variables are known in a given design point, this completely defines the geometry and other properties of the structure. In order to guarantee this, the chosen design variables must be independent to each other.

1.3.2. Optimization constraints

Some designs are useful solutions to the optimization problem, but others might be inadequate in terms of function, behaviour, or other considerations. If a design meets all the requirements placed on it, it will be called a feasible design. In most cases, the starting design is a feasible design. The restrictions that must be satisfied in order to produce a feasible design are called constraints.
From a physical point of view the constraints can be separated into two groups:

Constraints imposed on the design variables and which restrict their range for reasons other than behaviour considerations will be called design constraints or side constraints. The geometrical optimization constraints describes the lower the upper limit of the design variables. These are expressed in an explicit form of the design variables. These could be for example a minimum and maximum thickness of the plate.

Constraints that derive from behaviour requirements will be called behaviour constraints. Limitations on the maximum stresses, displacements, or buckling strength are typical examples

of behaviour constraints. This type of constraints based on a result of a structural analysis. Explicit and implicit behaviour constraints are both encountered in practical design. Explicit behaviour constraints are often given by formulas presented in design codes or specifications.

From the mathematical point of view, in most cases, constraints may usually be expressed as a set of inequalities:

$$g_j(x_i) \le 0 \quad (j=1,...,m \; ; \; i=1,...,n), \tag{1.1}$$

where m is the number of inequality constraints and x_i is the vector of design variables. In a structural design problem, one has also to consider equality constraints of the general form:

$$h_j(x_i) = 0 \quad (j=m+1,...,p), \tag{1.2}$$

where p-m is the number of equalities. In many cases equality constraints can be used to eliminate variables from the optimization process, thereby reducing their number.

The equality-type constraints can be used to reduce the number of design variables. This type of constraints may represent also various design considerations such as a desired ratio between the width of a cross section and its depth.

We may view each design variable as one dimension in a design space and any particular set of variables as a point in this space. In case of two design variables the design space reduces to a plan, but in the general case of n variables, we have an n-dimensional hyperspace. A design which satisfies all the constraints is a feasible design. The set of values of the design variables that satisfy the equation $g_j(x_i) = 0$ forms a surface in the design space. It is a surface in the sense that it cuts the space into two regions: where one where $g_j(x_i) > 0$ and the other $g_j(x_i)<0$. The design space and the constraint surfaces for the three-bar truss example are shown in Figure 1.2. The set of all feasible designs form the feasible region. Points on the surface are called constrained designs. The j^{th} constraint is said to be active in a design point for which $g_j(x_i) = 0$ and passive if $g_j(x_i) <0$. If $g_j(x_i) > 0$ the constraint is violated and the corresponding design is infeasible.

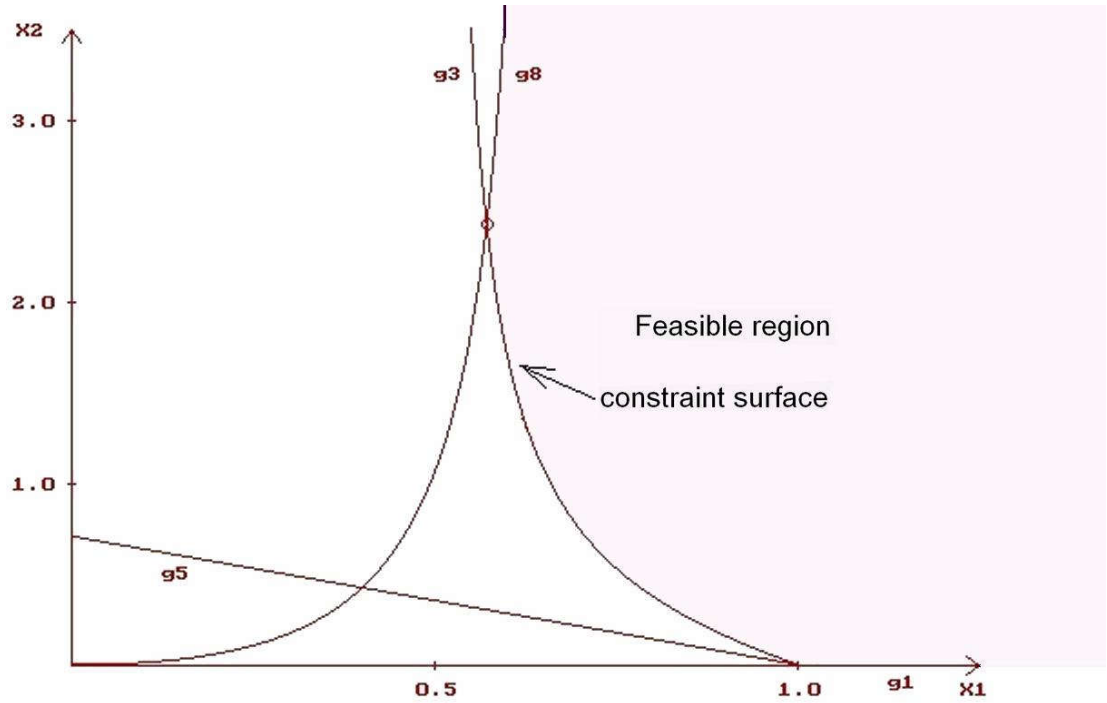

Figure 1.2.: Optimality constraints in the space of the design variables

1.3.3. The objective function

There usually exist an infinite number of feasible designs. In order to find the best one, it is necessary to form a function of the variables to use for comparison of feasible design alternatives. The objective function (or cost function) is the function whose least value is usually wanted in an optimization procedure. It is a function of the design variables and it may represent the weight, the cost of the structure, or any other criterion by which some possible designs are preferred to others. We always assume that the objective function ($Z = F(x_i)$), is to be minimized, which entails no loss of generality since the minimum of $-F(x_i)$ occurs where the maximum of $F(x_i)$ takes place (see Figure 1.3). The selecting the objective function has got a significant impact on the entire optimization process. For example, if the cost of the structure is assumed to be proportional to its weight, then the objective function will represent the weight. The weight of the structure is often of critical importance, but the minimum weight is not always the cheapest. In general, the objective function represents the most important single property of a design, but it may represent also a weighted sum of a number of properties. A general cost function may include the cost of materials, fabrication, transportation, operation, repair, and many other cost factors. In this case, large numbers of members are considered in the form of the objective function, where it is appropriate to analyze the impact of certain members of the product price. Special attention should be paid to the components, which can result a "nearly constant" objective function. They are not worth to take into account. It is not true, that the most complex objective function gives the best results. In general, the objective function is a nonlinear function of the design variables.

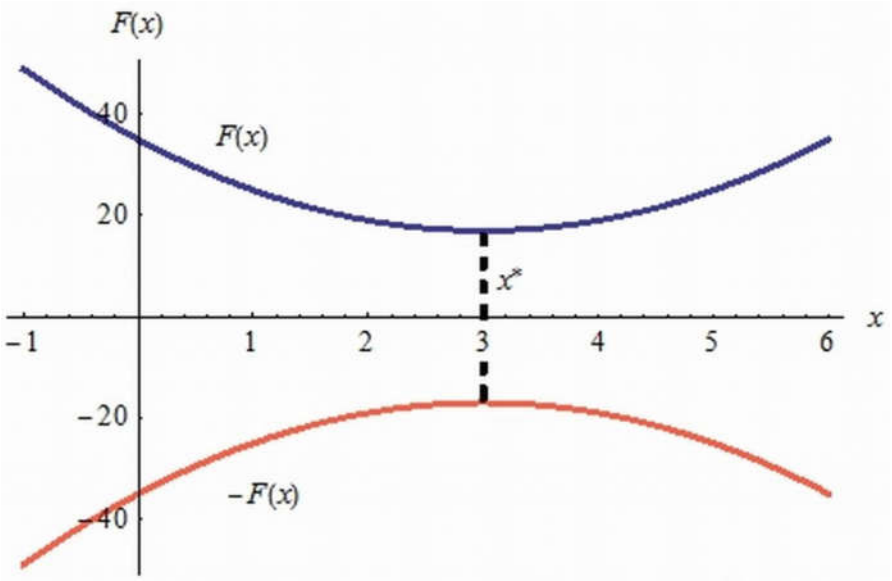

Figure 1.3.: The local extremum of the goal function

It is also possible to optimize simultaneously for multiple objective function, but this is only recommended if dominant objective function could not be selected, or the objective functions are in contradiction. This is the field of multi-objective optimization. The simplest solution technique is if we create a weighted sum of the objective functions and solve the problem as a standard optimization problem with only one objective function.
Pareto has developed the theory of the multi-objective optimization in 1896. It is important to note that the solution of an optimization problem with one goal function is generally a design point in the space of the design variables, while the solution of the Pareto-optimization problem is a set of design points, called Pareto front. A Pareto optimal solution is found if there is no other feasible solution that would reduce some objective function without causing a simultaneous increase in at least one other objective function.

We illustrated this complicated field with an example about „Optimization of Geometry for Car Bag" (this example was solved by company Sigma technology using the IOSO optimization software: www.iosotech.com), which can be found in the attached database for examples.

1.3.4. Formulating the optimization problem

The general formulation of the constrained optimization problem in the n dimensional Euclidian space is the following:

$$\begin{aligned} Z = F(x_i) &\rightarrow \min \\ g_j(x_i) \le 0 \; i = 1,\ldots,n \quad j = 1,\ldots,m \\ h_j(x_i) = 0 \; j = m+1,\ldots,p\,. \end{aligned} \tag{1.3}$$

This is an optimization problem with one goal function ($F(x_i)$), with inequality ($g(x_i)$) and equality ($h(x_i)$) constraints is formulated for n design variable. The numbers of the inequality

constraints are m and the numbers of the equality constraints are (*p-m*). In the engineering problem definition, we often highlight the side constraints, defining the searching domain in the *n* dimensional space. This could be formulated as follows:

$$\check{x}_i \le x_i \le \hat{x}_i \quad i = 1,\ldots,n \ , \tag{1.4}$$

where $\check{x}_i$ and $\hat{x}_i$ are the lower and upper limits of the searching domain.

1.4. Optimization examples

Large number of the optimization problem types made impossible to introduce all relevant fields, as optimization problems can be formulated on every side of the life, they are not only related to engineering problems.

The field of economics also often uses the optimization methods for performing economic analysis and supporting decision. Maximizing the profit or optimizing the elements of a given stock portfolio is an important task.
In the production technology the maximum capacity utilization of the different machines plays a very important role, considering different raw materials, which is not available in unlimited quantity.

The known travelling salesman is also an optimization problem: finding the shortest way between the cities. The solution for 50 cities can be seen on Figure 1.4.

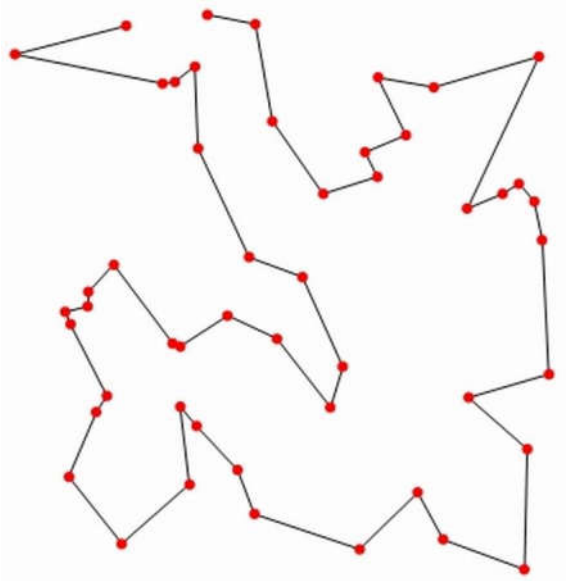

Figure 1.4.: The solution of the travelling salesman problem for 50 cities (Wolfram Mathematica)

In the area of the image and audio processing, for example, an optimization problem is to reduce the noise. A large field of the mathematics is the game theory, uses the optimization techniques very intensively too.

In the mechanical engineering practice the first industrial applications were in the field of military and space industry. In the Figure 1.5 the outline of a modern military airplane (F22) can be seen, by the design, beside the minimum weight, also the minimization of the radar cross section was an important task. In the space industry the weight minimization problem has got a very important role. An example in the field of space industry is a weight minimization of support in a rocket, which includes both topology and shape optimization problem (see Figure 1.6).

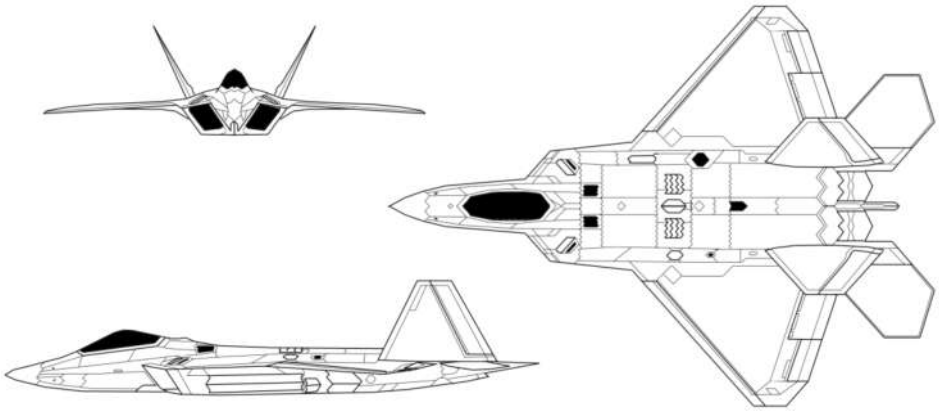

Figure 1.5.: Military airplane

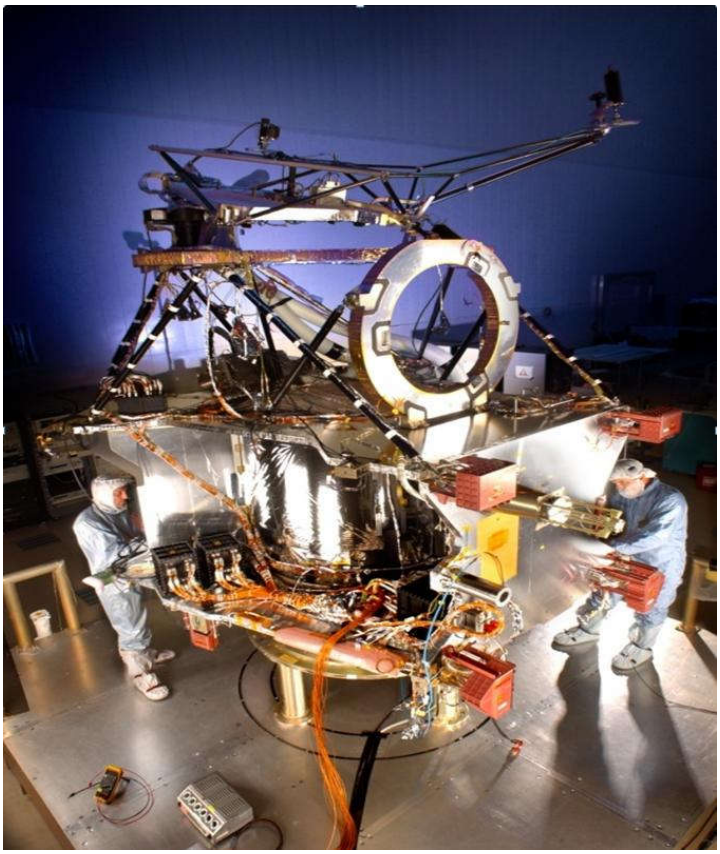

Figure 1.6.: Support in a rocket

Designing trucks, buses and cars different optimization techniques were used also very intensively. Nowadays most parts of these vehicles are undergone some level of optimization process even if the weight of the part is neglectable comparing the weight of the whole car. Because of the heavy price competition by the suppliers in the car industry, the 5-10% weight reduction which can be reached by using shape optimization techniques, gives significant advantages to the company.
Nowdays several optimization processes can be combined with different structural analysis modules. Not only with static load cases can be considered, but for example dynamic, temperature, flow, nonlinear systems can be optimized too. Additionally not only parts, but whole subassemblies using contact conditions between the parts can be solved.
But we have to know, that because the optimizer call lot of times the structural analysis module, the complicated problems often very time consuming if we have only one PC. Generally this type of problem formulation leads to the field of supercomputing; so supercomputers or computer clusters should be used to solve such type of industrial problems.

1.5. References

[1.1] Jármai Károly, Iványi Miklós: Gazdaságos fémszerkezetek analízise és tervezése. Műegyetemi kiadó, 2001

1.6. Questions

1. Define the general form of an optimization problem!
2. Define the design variable, design parameter and goal function!
3. What kind of classes are avaiable as design variables from phisical and mathematical point of view?
4. What is the main difference between the side constraints and behaviour constraints?
5. Explain some relevant optimization examples from the different field of the industry?

2. SINGLE-VARIABLE OPTIMIZATION

Only single-variable functions are considered in this chapter. These methods will be used in later chapters for multi-variable function optimization.

2.1. Optimality Criteria

Sufficient conditions of optimality:
Suppose at point x^*, the first derivative is zero and the first nonzero higher order derivative is denoted by n

(i) If n is odd, x^* is a point of inflection
(ii) If n is even, x^* is a local optimum
(a) If that derivative is positive, x^* is a local minimum
(b) If that derivative is negative, x^* is a local maximum.

2.2. Bracketing Algorithms

Finds a lower and an upper bound of the minimum point

2.2.1. Exhaustive Search

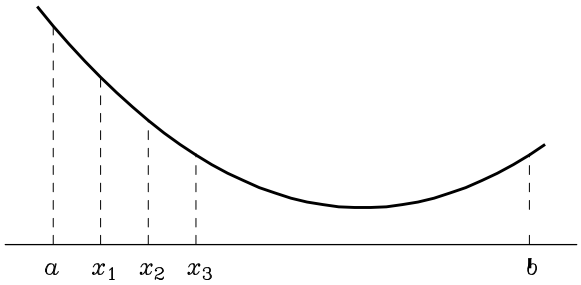

Figure 2.1 The exhaustive search method uses equally spaced points.

Function values are evaluated at n equally spaced points (Figure 2.1).

Algorithm:

Step 1: Set $x^{(0)} = a$, $x^{(n+1)} = b$, $x^{(j)} = a + \frac{b-a}{n+1} j$, $1 \le j \le n$. Set $k = 1$.

Step 2: If $f\left(x^{(k-1)}\right) \ge f\left(x^{(k)}\right) \le f\left(x^{(k+1)}\right)$, the minimum lies in $\left(x^{(k-1)}, x^{(k+1)}\right)$, **Terminate.**
Else go to Step 3.

Step 3: Is $k = n$? If no, set $k = k + 1$, go to Step 2.

If yes, no minimum exists in (a, b) Accuracy in the result: $\frac{2}{n+1}(b-a)$.

Average number of function evaluations to get to the optimum is $\left(\frac{n}{2} + 2\right)$.

2.2.2. *Bounding Phase Method*

This method is used to bracket the minimum of a function.
Algorithm:

Step 1: Choose an initial guess $x^{(0)}$ and an increment Δ. Set $k=0$.

Step 2: If $f\left(x^{(0)}-|\Delta|\right)\geq f\left(x^{(0)}\right)\geq f\left(x^{(0)}+|\Delta|\right)$ then Δ is positive.

Else if $f\left(x^{(0)}-|\Delta|\right)\leq f\left(x^{(0)}\right)\leq f\left(x^{(0)}+|\Delta|\right)$ then Δ is negative.

Else go to Step 1.

Step 3: Set $x^{(k+1)}=x^{(k)}+2^k\Delta$.

Step 4: If $f\left(x^{(k+1)}\right)<f\left(x^{(k)}\right)$, set $k=k+1$ and go to Step 3.

Else The minimum lies in the interval $\left(x^{(k-1)},x^{(k+1)}\right)$ and **Terminate.**

If Δ is large, poor bracketing.
If Δ is small, more evaluations.

2.3. Region-Elimination Methods

Fundamental rule for Region-elimination methods:
For $x_1<x_2$ where x_1, x_2 lie in (*a, b*).
1. If $f(x_1)>f(x_2)$ then minimum does not lie in (a,x_1).
2. If $f(x_1)<f(x_2)$ then minimum does not lie in (x_2,b).

2.3.1. *Golden Section Search*

Interval is reduced according to *golden* rule.
Properties: For only two trials, spread them equidistant from the center. Subinterval eliminated should be of the same length regardless of the outcome of the trial. Only one new point at each step is evaluated, other point remains from the previous step.

Algorithm:

Step 1: Choose a lower bound *a* and an upper bound *b*. Also choose a small number ε. Normalize the variable *x* by using the equation, $\omega=(x-a)/(b-a)$. Thus, $a_\omega=0$, $b_\omega=1$, and $L_\omega=1$. Set $k=1$.

Step 2: Set $\omega_1=a_\omega+(0.618)L_\omega$ and $w_2=b_\omega-(0.618)L_\omega$. Compute $f(\omega_1)$ or f $f(\omega_2)$ depending on whichever was not evaluated earlier. Use the fundamental region-elimination rule to eliminate a region. Set new a_ω and b_ω.

Step 3: Is $|L_\omega|<\varepsilon$ small? If no, set $k=k+1$, go to Step 2. If yes, **Terminate.**

Interval reduces to $(0.618)^{n-1}$ after n evaluations. One new function evaluation at each iteration.
Consider the following function:

$$f(x) = x^2 + 54/x$$

Step 1: We choose $a = 0$ and $b = 5$. The transformation equation becomes $\omega = x/5$. Thus, $a_\omega = 0$, $b_\omega = 1$, and $L_\omega = 1$. Since the golden section method works with a transformed variable ω, it is convenient to work with the transformed function:

$$f(\omega) = 25\omega^2 + 54/(5\omega)$$

In the ω-space, the minimum lies at $\omega^* = 3/5 = 0.6$. We set an iteration counter $k = 1$.

Step 2: We set $\omega_1 = 0 + (0.618)1 = 0.618$ and $\omega_2 = 1 - (0.618)1$ or $\omega_2 = 0.382$. The corresponding function values are $f(\omega_1) = 27.02$ and $f(\omega_2) = 31.92$. Since $f(\omega_1) < f(\omega_2)$, the minimum cannot lie in any point smaller than $\omega_2 = 0.382$. Thus, we eliminate the region (a, ω_2) or (0, 0.382). Thus, $a_\omega = 0.382$ and $b_\omega = 1$. At this stage, $L_\omega = 1 - 0.382 = 0.618$. The region being eliminated after this iteration is shown in Figure 2.2. The position of the exact minimum at $\omega = 0.6$ is also shown.

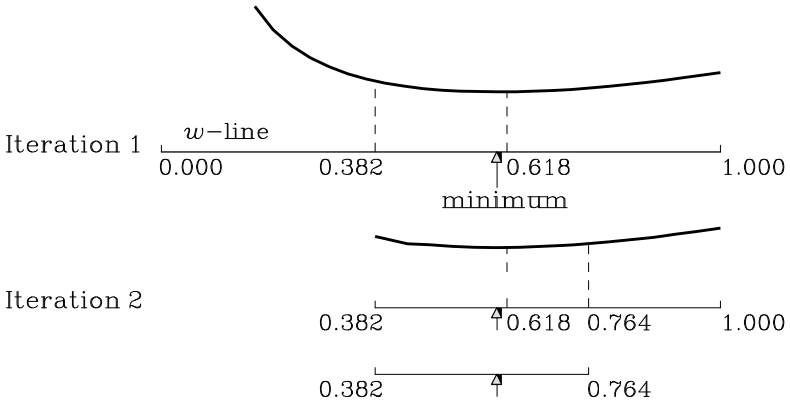

Figure 2.2: Region eliminations in the first two iterations of the golden section search algorithm.

Step 3: Since $|L_\omega|$ is not smaller than ε, we set $k = 2$ and move to Step 2. This completes one iteration of the golden section search method.

Step 2: For the second iteration, we set

$$\omega_1 = 0.382 + (0.618)0.618 = 0.764$$
$$\omega_2 = 1 - (0.618)0.618 = 0.618.$$

We observe that the point ω_2 was computed in the previous iteration. Thus, we only need to compute the function value at ω_1: $f(\omega_1) = 28.73$. Using the fundamental region-elimination rule and observing the relation $f(\omega_1) > f(\omega_2)$, we eliminate the interval (0.764, 1). Thus, the

new bounds are $a_\omega = 0.382$ and $b_\omega = 0.764$, and the new interval is $L_\omega = 0.764 - 0.382 = 0.382$, which is incidentally equal to $(0.618)^2$! Figure 2.2 shows the final region after two iterations of this algorithm.

Step 3: Since the obtained interval is not smaller than ε, we continue to proceed to Step 2 after incrementing the iteration counter k to 3.

Step 2: Here, we observe that $\omega_1 = 0.618$ and $\omega_2 = 0.528$, of which the point ω_1 was evaluated before. Thus, we compute $f(\omega_2)$ only: $f(\omega_2) = 27.43$. We also observe that $f(\omega_1) < f(\omega_2)$ and we eliminate the interval (0.382, 0.528). The new interval is (0.528, 0.764) and the new range is $L_\omega = 0.764 - 0.528 = 0.236$, which is exactly equal to $(0.618)^3$!

Step 3: Thus, at the end of the third iteration, $L_\omega = 0.236$. This way, Steps 2 and 3 may be continued until the desired accuracy is achieved.

We observe that at each iteration, only one new function evaluation is necessary. After three iterations, we have performed only four function evaluations. Thus, the interval reduces to $(0.618)^3$ or 0.236.

2.4. Methods Requiring Derivatives

- Use gradient information.
- At local minimum, the derivative of the function is equal to zero.

Gradients are computed numerically as follows:

$$f'(x^{(t)}) = \frac{f(x^{(t)}+\Delta x^{(t)}) - f(x^{(t)}-\Delta x^{(t)})}{2\Delta x^{(\neq)}} \tag{2.1}$$

$$f''(x^{(t)}) = \frac{f(x^{(t)}+\Delta x^{(t)}) - 2f(x^{(t)}) + f(x^{(t)}-\Delta x^{(t)})}{(\Delta x^{(t)})^2} \tag{2.2}$$

The parameter $\Delta x^{(t)}$ is usually taken to be a small value. In all our calculations, we assign $\Delta x^{(t)}$ to be about 1 per cent of $x^{(t)}$:

$$\Delta x^{(t)} = \begin{cases} 0.01|x^{(t)}| & if \ |x^{(t)}| > 0.01 \\ 0.0001 & otherwise. \end{cases} \tag{2.3}$$

According to Equations (2.1) and (2.2), the first derivative requires two function evaluations and the second derivative requires three function evaluations.

2.4.1. Newton-Raphson Method

A linear approximation to the derivative is made to zero to derive the transition rule.

Algorithm:

Step 1: Choose initial guess x_1 and a small number ε. Set $k = 1$. Compute $f'(x_1)$.

Step 2: Compute $f''(x_k)$.

Step 3: Calculate $x_{k+1} = x_k - \dfrac{f'(x_k)}{f''(x_k)}$. Compute $f'(x_{k+1})$.

Step 4: If $|f'(x_{k+1})| < \varepsilon$, **Terminate.**

Else set $k = k+1$ and Go to step 2.

Convergence of the algorithm depends on the initial point and the nature of the objective function.

2.4.2. Bisection Method

Both function value and sign of derivative are used to derive the transition rule.

Algorithm:

Step 1: Choose two points a and b such that $f'(a) < 0$ and $f'(b) > 0$, and a small number ε. Set $L = a$ and $R = b$.

Step 2: Calculate $z = (R+L)/2$ and evaluate $f'(z)$.

Step 3: If $|f'(z)| \le \varepsilon$, **Terminate.**

Else if $f'(z) < 0$ set $L = z$ and Go to step 2; else if $f'(z) > 0$ set $R = z$ and Go to step 2.

Consider again the function:

$$f(x) = x^2 + 54/x.$$

Step 1: We choose two points $a = 2$ and $b = 5$ such that $f'(a) = -9.501$ and $f'(b) = 7.841$ are of opposite sign. The derivatives are computed numerically using Equation (2.1). We also choose a small number $\varepsilon = 10^{-3}$.

Step 2: We calculate a quantity $z = (x_1 + x_2)/2 = 3.5$ and compute $f'(z) = 2.591$.

Step 3: Since $f'(z) > 0$, the right-half of the search space needs to be eliminated. Thus, we set $x_1 = 2$ and $x_2 = z = 3{,}5$. This completes one iteration of the algorithm. At each iteration, only half of the search region is eliminated, but here the decision about which half to delete depends on the derivatives at the mid-point of the interval.

Step 2: We compute $z = (2 + 3.5)/2 = 2.750$ and $f'(z) = -1.641$.

Step 3: Since $f'(z) < 0$, we set $x_1 = 2.750$ and $x_2 = 3.500$.

Step 2: The new point z is the average of the two bounds: $z = 3.125$. The function value at this point is $f'(z) = 0.720$.

Step 3: Since $|f'(z)| \not< \varepsilon$, we continue with Step 2.

Thus, at the end of 10 function evaluations, we have obtained an interval (2.750, 3.125), bracketing the minimum point $x^* = 3.0$. The guess of the minimum point is the mid-point of the obtained interval or $x = 2.938$. This process continues until we find a point with a vanishing derivative. Since at each iteration, the gradient is evaluated only at one new point, the bisection method requires two function evaluations per iteration. In this method, exactly

half the region is eliminated at every iteration; but using the magnitude of the gradient, a faster algorithm can be designed to adaptively eliminate variable portions of search region – a matter which we discuss in the following subsection.

2.4.3. Secant Method

Both function value and derivative are used to derive the transition rule.
Algorithm:

Same as Bisection method except step 2 is modified as follows:

Step 2: Calculate $z = R - \dfrac{f'(R)}{\big(f'(R) - f'(L)\big)/(R-L)}$ and evaluate $f'(z)$.

2.5. References

[2.1] Powell, M. J.D. (1964): **An efficient method for finding the minimum of a function of several variables without calculating derivatives.** Computer Journal. 7, 155-162.

[2.2] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983): **Engineering Optimization-Methods and Applications**. New York: Wiley.

[2.3] Scarborough, J. B. (1966): **Numerical Mathematical Analysis.** New Delhi: Oxford & IBH Publishing Co.

2.6. Questions

1. Explain the sufficient conditions of the optimality!
2. Summarize the main characteristics of the bracketing methods!
3. Explain the fundamental rules of region-elimination methods!
4. What are the advantages of methods using derivatives?

3. MULTI-VARIABLE OPTIMIZATION

Functions of multiple variables (N variables) are considered for minimization here. Duality principle can be used to apply these methods to maximization problems.

3.1. Optimality Criteria

A stationary point $\bar{x}$ is minimum, maximum, or saddle-point if $\nabla^2 f(\bar{x})$ is positive definite, negative definite, or $\nabla^2 f(\bar{x}) \leqq\geqq 0$.

Necessary Conditions: For x^* to be a local minimum, $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

Sufficient Conditions: $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite than, x^* is an isolated local minimum of $f(x)$.

3.2. Direct Search Methods

Use only function values; no gradient information is used.

3.2.1. Simplex Search Method

- A simplex is a set of $(N+1)$ points. At each iteration, a new simplex is created from the current simplex by fixed transition rules.
- The worst point is projected a suitable distance through the centroid of the remaining points.

Algorithm:

Step 1: Choose $\gamma > 1$, $\beta \in (0, 1)$, and a termination parameter ε. Create an initial simplex[1].

Step 2: Find x_h (the worst point), x_l (the best point), and x_g (next to the worst point). Calculate $x_c = \frac{1}{N} \sum_{i=1, i\neq h}^{N+1} x_i$.

$$x_j^{(i)} = \begin{cases} x_j^{(0)} + C & if\ j = i; \\ x_j^{(0)} + C\ \Delta, & otherwise, \end{cases} \quad \text{where } \Delta = \begin{cases} 0.25, & if\ N = 3; \\ \left(\frac{\sqrt{N+J}-2}{N-3}\right) & otherwise. \end{cases}$$

Step 3: Calculate the reflected point $x_r = 2x_c - x_h$. Set $x_{new} = x_r$.

If $f(x_r) < f(x_l)$, set $x_{new} = (1+\gamma)x_c - \gamma x_h$ (expansion).

Else if $f(x_r) \geq f(x_h)$, set $x_{new} = (1-\beta)x_c - \beta x_h$ (contraction).

---

[1] One of the ways to create a simplex is to choose a base point x^0 and a scale factor C. Then $(N+1)$ points are $x^{(0)}$ and for $i, j = 1, 2, \ldots, N$.

Else if $f(x_g) < f(x_r) < f(x_h)$, set $x_{new} = (1+\beta)x_c - \beta x_h$ (contraction).
Calculate $f(x_{new})$ and replace x_h by x_{new}.

Step 4: If $\left\{\sum_{i=1}^{N+1}\frac{(f(x_i)-f(x_c))^2}{N+1}\right\}^{1/2} \le \varepsilon$, **Terminate**.

Else go to Step 2.

3.2.2. Powell's Conjugate Direction Method

- Most successful direct search method
- Uses history of iterations to create new search directions
- Based on a quadratic model
- Generate N conjugate directions and perform one-dimensional search in each direction one at a time

Parallel Subspace Property: Given a quadratic function $q(x)$, two arbitrary but distinct points $x^{(1)}$ and $x^{(2)}$, and a direction d. If $y^{(1)}$ is the solution to min $q(x^{(1)}+\lambda d)$ and $y^{(2)}$ is the solution to min $q(x^{(2)}+\lambda d)$, then the direction $(y^{(2)}-y^{(1)})$ is C-conjugate to d or $(y^{(2)}-y^{(1)})^T Cd =$ Diagonal matrix.

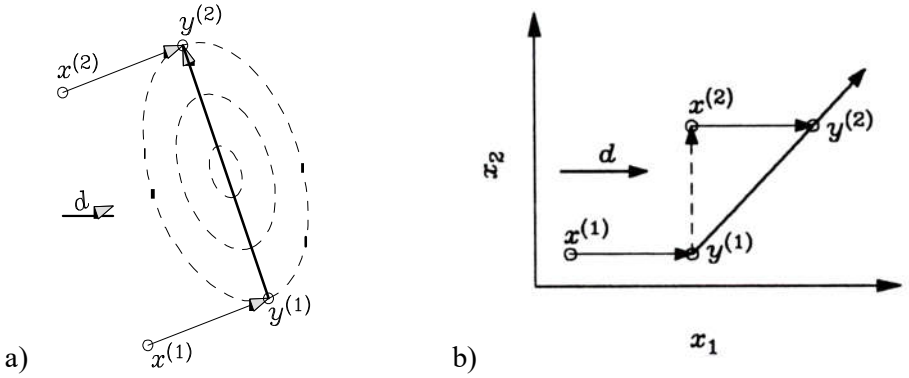

Figure 3.1 Illustration of the paralell subspace property with two arbitrary points and an arbitrary search direction in (a). The same can also be achieved from one point and two coordinate points in (b).

Instead of using two points and a direction vector to create one conjugate direction, one point and coordinate directions can be used to create conjugate directions (Figure 3.1).

Extended Parallel Subspace Property: In higher dimensions, if from $x^{(1)}$ the point $y^{(1)}$ is found after searches along each of $m(<n)$ conjugate directions, and similarly if from $x^{(2)}$ the point $y^{(2)}$ is found after searches along each of m conjugate directions, $s^{(1)}$, $s^{(2)}$,..., $s^{(m)}$, then the vector $(y^{(2)}-y^{(1)})$ will be the conjugate to all of the m previous directions.

Algorithm:

Step 1: Choose a starting point $x^{(0)}$ and a set of N linearly independent directions; possibly $s^{(i)} = e^{(i)}$ for i=1, 2,…, N.

Step 2: Minimize along N unidirectional search directions using the previous minimum point to begin the next search. Begin with the search $s^{(1)}$ direction and end with $s^{(N)}$. Thereafter, perform another unidirectional search along $s^{(1)}$.

Step 3: Form a new conjugate direction d using the extended parallel subspace property.

Step 4: If $\|d\|$ is small or search directions are linearly independent, **Terminate.**
Else replace $s^{(j)} = s^{(j-1)}$ for all $j = N, N-1, ..., 2$. Set $s^{(1)} = d/\|d\|$ and go to Step 2.

A test is required to ensure linear independence of conjugate directions. If the function is quadratic, exactly N loops through steps 2 to 4 are required. If the function is quadratic, exactly $(N-1)$ loops through Steps 2 to 4 is required. Since in every iteration of the above algorithm exactly $(N+1)$ unidirectional searches are necessary, a total of $(N-1)\times(N+1)$ or (N^2-1) unidirectional searches are necessary to find N conjugate directions. Thereafter, one final unidirectional search is necessary to obtain the minimum point. Thus, in order to find the minimum of a quadratic objective function, the conjugate direction method requires a total of N^2 unidirectional searches.
Disadvantages:
- It takes usually more than N cycles for nonquadratic functions
- One-dimensional searches may not be exact, so directions may not be conjugate
- May halt before the optima is reached

Consider the Himmelblau function:

$$\text{Minimize } f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

in the interval $0 \le x_1, x_2 \le 5$.

Step 1: We begin with a point $x^{(0)} = (0, 4)^T$. We assume initial search directions as $s^{(1)} = (1, 0)^T$ and $s^{(2)} = (0, 1)^T$.

Step 2: We first find the minimum point along the search direction $s^{(1)}$. Any point along that direction can be written as $x^p = x^{(0)} + \alpha s^{(1)}$, where α is a scalar quantity expressing the distance of the point x^p from $x^{(0)}$. Thus, the point x^p can be written as $x^p = (\alpha, 4)^T$. Now the two-variable function $f(x_1, x_2)$ can be expressed in terms of one variable α as

$$F(\alpha) = (\alpha^2 - 7)^2 + (\alpha + 9)^{2_1} + x_2^2 - 7)^2,$$

which represents the function value of any point along the direction $s^{(1)}$ and passing through $x^{(0)}$. Since we are looking for the point for which the function value is minimum, we may differentiate the above expression with respect to α and equate to zero. But in any arbitrary problem, it may not be possible to write an explicit expression of the single-variable function $F(\alpha)$ and differentiate. In those cases, the function $F(\alpha)$ can be obtained by substituting each variable x_i by x_i^p. Thereafter, any single-variable optimization methods, as described in Chapter 2, can be used to find the minimum point. The first task is to bracket the minimum and then the subsequent task is to find the minimum point. Here, we could have found the exact minimum solution by differentiating the single-variable function $F(\alpha)$ with respect to α and then equating the term to zero, but we follow the more generic procedure of numerical differentiation, a method which will be used in many real-world optimization problems. Using the bounding phase method in the above problem we find that the minimum is bracketed in the interval (1,4) and using the golden section search we obtain the minimum $\alpha^* = 2.083$ with three decimal places of accuracy. Thus, $x^{(1)} = (2.083, 4.000)^T$.

Similarly, we find the minimum point along the second search direction $s^{(2)}$ from the point $x^{(1)}$. A general point on that line is

$$x(\alpha) = x^{(1)} + \alpha s^{(2)} = (2.083.\ (4 + \alpha))^{\mathrm{T}}.$$

The optimum point found using a combined application of the bounding phase and the golden section search method is $\alpha^* = -1{,}592$ and the corresponding point is $x^{(2)} = (2.083, 2.408)^T$.

From the point $x^{(2)}$, we perform a final unidirectional search along the first search direction and obtain the minimum point $x^{(3)} = (2.881, 2.408)^T$.

Step 3: According to the parellel subspace property, we find the new conjugate direction.

$$d = x^{(3)} - x^{(1)} = (2.881, 2.408\ )^{\mathrm{T}} - (2.083{,}4.000\ )^{\mathrm{T}} = (0.798{-}1.592)^T$$

Step 4: The magnitude of search vector d is not small. Thus, the new conjugate search directions are

$$\begin{aligned} s^{(2)} &= (1,\ 0)^T, \\ s^{(1)} &= (0.798, -1.592)^T / \|(0.798, -1.592)^T\| \\ &= (0.448, -0.894)^T. \end{aligned}$$

This completes one iteration of Powell's conjugate direction method. Figure 3.2 shows the new conjugate direction on a contour plot of the objective function. With these new search directions we now proceed to Step 2.

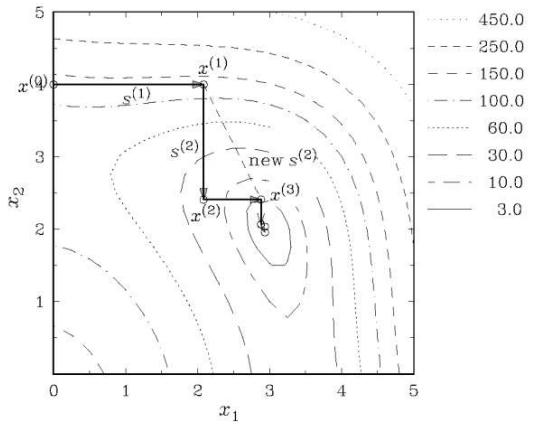

Figure 3.2 Two iterations of Powell's conjugate direction method.

Step 2: A single-variable minimization along the search direction $s^{(1)}$ from the point $x^{(3)} = (2.881, 2.408)^T$ results in the new point $x^{(4)} = (3.063, 2.045)^T$. If the objective function had been a quadratic function, we would have achieved the optimum point at this step. However, it is interesting to note that the solution $x^{(4)}$ is close to the true minimum of the function. One more unidirectional search along $s^{(2)}$ from the point $x^{(4)}$ results in the point $x^{(5)} = (2.988, 2.045)^T$. Another minimization along $s^{(1)}$ results in $x^{(6)} = (3.008, 2.006)^T$ $f(x^{(6)} = 0.004)$.

Step 3: The new conjugate direction is

$$d = (x^{(6)} - x^{(4)}) = (0.055, -0.039)^T$$

The unit vector along this direction is $(0.816, -0.578)^{\mathrm{T}}$.

Step 4: The new pair of conjugate search directions are $s^{(1)} = (0.448, -0.894)^T$ and $s^{(2)} = (0.055, -0.039)^T$ respectively. The search direction *d* (before normalizing) may be considered to be small and therefore the algorithm may terminate.

We observe that in one iteration of Step 2, $(N+1)$ unidirectional searches are necessary. Thus, computationally this method may be expensive. In terms of the storage requirement, the algorithm has to store $(N+1)$ points and *N* search directions at any stage of the iteration.

3.3. Gradient-based Methods

- Direct search methods are expensive
- Gradient-based methods assume existence of $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$
- All methods employ

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s(x^{(k)})$$

where $\alpha^{(k)}$ is the step length and $s(x^{(k)})$ is the search direction.

– Usually, $\alpha^{(k)}$ is selected to minimize the function in $s(x^{(k)})$ direction. Update search direction iteratively.

3.3.1. Numerical Gradient Approximation

- In real-world engineering problems, gradients are often difficult to calculate.
- Even if they are available, their computation could be error prone
- Numerically gradients are computed:

$$\frac{\partial f(x)}{\partial x_i}\Big|_{(x)^t} = (f(x_i^{(t)} + \Delta x_i^{(t)}) - f(x_i^{(t)} - \Delta x_i^{(t)}))/2\,\Delta x_i^{(t)}$$

$$\frac{\partial^2 f(x)}{\partial^2 x_i^2}\Big|x^{(t)} = (f(x_i^{(t)} + \Delta x_i^{(t)}) - 2f(x^{(t)}) + f(x_i^{(t)} - \Delta x_i^{(t)}))/(\Delta x_i^{(t)})^2.$$

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\Big|x^{(t)} = [f(x_i^{(t)} + \Delta x_{i)}^{(t)} x_j^{(t)} + \Delta x_j^{(t)})$$

$$-f(x_i^{(t)} + \Delta x_{i)}^{(t)} x_j^{(t)} - \Delta x_j^{(t)}) - f(x_i^{(t)} - \Delta x_{i)}^{(t)} x_j^{(t)} + \Delta x_j^{(t)})$$

$$+f(x_i^{(t)} - \Delta x_{i)}^{(t)} x_j^{(t)} - \Delta x_j^{(t)})]\,/(4\,\Delta x_i^{(t)}\,\Delta x_j^{(t)}).$$

The computation of the first derivative with respect to each variable requires two function evaluations, thus totaling $2N$ function evaluations for the complete first derivative vector. The computation of the second derivative $\partial^2 f / \partial x_i^2$ requires three function evaluations, but the second-order partial derivative $\partial^2 f /(\partial x_i, \partial x_j)$ requires four function evaluations. Thus, the computation of Hessian matrix requires $(2N^2 + 1)$ function evaluations (assuming the symmetry of the matrix).

3.3.2. Cauchy's Method (Steepest Descent)

Greatest decrease in $s(x^{(k)}) = -\nabla f(x)$ direction.

Algorithm:

Step 1: Choose a maximum number of iterations M to be performed, an initial point $x^{(0)}$, two termination parameters $\varepsilon_1, \varepsilon_2$, and set $k = 0$.

Step 2: Calculate $\nabla f(x^{(k)})$, the first derivative at the point $x^{(k)}$.

Step 3: If $\left\|\nabla f(x^{(k)})\right\| \le \varepsilon_1$, **Terminate.**

Else if $k \ge M$, **Terminate.**
Else go to Step 4.

Step 4: Perform a unidirectional search to find $\alpha^{(k)}$ using ε_2 such that /($f(x^{(k+1)}) = f(x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}))$ is minimum. One criterion for termination is when $\left\| \nabla f(x^{(k+1)}) \cdot \nabla f(x^{(k)}) \right\| \le \varepsilon_2$.

Step 5: Is $\left\| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} \right\| \le \varepsilon_1$? If yes, **Terminate.**

Else set $k = k + 1$ and go to Step 2.

Cauchy's method works well when $x^{(0)}$ is far away from x^*.

3.3.3. Newton's Method

Use a search direction $s(x^{(k)}) = -\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)})$.

Algorithm: Same as Cauchy's method except step 4 is modified:

Step 4: Perform a line search to find $\alpha^{(k)}$ using ε_2 such that $f(x^{(k+1)}) = f(x^{(k)} - \alpha^{(k)} \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}))$ is minimum.

Newton's method works well when $x^{(0)}$ is close to x^*.

3.3.4. Marquardt's Method

- A compromise between Cauchy's and Newton's method.
- When the point is far away from the optimum, Cauchy's method is used and when the point is close to the optimum Newton's method is used.

Algorithm:

Step 1: Choose a starting point, $x^{(0)}$, the maximum number of iterations, M, and a termination parameter, ε. Set $k = 0$ and $\lambda^{(0)} = 10^4$ (a large number).

Step 2: Calculate $\nabla f(x^{(k)})$.

Step 3: If $\left\| \nabla f(x^{(k)}) \right\| \le \varepsilon$ or $k \ge M$, **Terminate.**

Else go to Step 4

Step 4: Calculate $s(x^{(k)}) = -\left[H^{(k)} + \lambda^{(k)} I \right]^{-1} \nabla f(x^{(k)})$.

Set $x^{(k+1)} = x^{(k)} + s(x^{(k)})$.

Step 5: Is $f(x^{(k+1)}) < f(x^{(k)})$? If yes, go to Step 6.

Else go to Step 7.

Step 6: Set $\lambda^{(k+1)} = \frac{1}{2} \lambda^{(k)}$, $k = k + 1$, and go to Step 2.

Step 7: Set $\lambda^{(k)} = 2\lambda^{(k)}$ and go to Step 4.

Need to compute Hessian matrix, which may be cumbersome.

3.3.5. Variable-Metric Method (Davidon-Fletcher-Powell method)

- Uses positive characteristics of Newton's method using only first-order information
- The search direction is

$$S(x^{(k)}) = -A^{(k)}\nabla f(x^{(k)}) \tag{3.1}$$

where $A^{(k)}$ is the Hessian matrix.

– Starting with an identity matrix $(A^{(0)} = I)$, iteratively evaluate the Hessian matrix with $g(x^{(k)}) = \nabla f(x^{(k)})$:

$$A^{(k)} = A^{(k-1)} + \frac{\Delta x^{(k-1)} \Delta x^{(k-1)^T}}{\Delta x^{(k-1)^T} \Delta e(x^{(k-1)})}$$

$$- \frac{A^{(k-1)} \Delta e(x^{(k-1)})(A^{(k-1)} \Delta e(x^{(k-1)}))^T}{\Delta e(x^{(k-1)})^\tau A(k-1) \Delta e(x^{(k-1)})} \tag{3.2}$$

– The above relation preserves the positive definiteness of the matrix A.

Algorithm:

Same as Fletcher-Reeves algorithm except the expression for search direction $s(x^k)$ in step 4, which is set according to equation 3.1.

Let us consider the Himmelblau function again:

$$\text{Minimize } f(x_1 x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

The inverse of the Hessian matrix at the minimum point $x^* = (3, 2)^T$ is found to be

$$H^{-1}(x^*) = \begin{pmatrix} 0.016 & -0.009 \\ -0.009 & 0.035 \end{pmatrix}. \tag{3.3}$$

Successive iterations of the DFP method transform an initial identity matrix into the above matrix. This allows the DFP search to become similar to Newton's search after a few iterations. The advantage of this method over Newton's method is that the inverse of the Hessian matrix need not be calculated. Thus, the algorithm has the effect of a second-order search, but the search is achieved only with first-order derivatives.

Step 1: Once again we begin with the initial point $x^{(0)} = (0, 0)^T$. The termination parameters are all set to be 10^{-3}.

Step 2: The derivative vector at the initial point is equal to $\nabla f(x^{(0)}) = (-14, -22)^T$. The search direction is $s^{(0)} = (14, 22)^T$.

Step 3: A unidirectional search from $x^{(0)}$ along $s^{(0)}$ gives the minimum point. We calculate the gradient at this point:

$$\nabla f(x^{(1)}) = (-30.707, 18.803)^T$$

In order to calculate the new search direction, we first compute the parameters required to be used in Equation (3.2):

$$\Delta x^{(0)} = x^{(1)} - x^{(0)} = (1.788, 2.810)^T,$$
$$\Delta e(x^{(0)}) = e(x^{(1)}) - e(x^{(0)}) = (-16.707, 40.803)^T$$
$$A^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

With these parameters, we compute the next estimate of the matrix A as follows:

$$\begin{aligned} A^{(1)} &= A^{(0)} + \frac{\Delta x^{(0)} \Delta x^{(0)^T}}{\Delta x^{(0)^T} \Delta e(x^{(0)})} - \frac{A^{(0)} \Delta e(x^{(0)}) (A^{(0)} \Delta e(x^{(0)}))^T}{\Delta e(x^{(0)})^T A^{(0)} \Delta e(x^{(0)})}, \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{\begin{pmatrix} 1.778 \\ 2.810 \end{pmatrix} (1.788,\ 2.810)}{(1.788, 2.810) \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix}} \\ &\quad - \frac{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix} \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix} \right)^T}{(-16.707, 40.803) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -16.707 \\ 40.803 \end{pmatrix}}, \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0.038 & 0.060 \\ 0.060 & 0.093 \end{pmatrix} - \begin{pmatrix} 0.144 & -0.351 \\ -0.351 & 0.856 \end{pmatrix}, \end{aligned}$$

Knowing $A^{(1)}$, we now obtain the search direction $s^{(1)}$ using Equation (3.1) as follows:

$$s^{(1)} = -\begin{pmatrix} 0.894 & 0.411 \\ 0.411 & 0.237 \end{pmatrix} \begin{pmatrix} -30.707 \\ 18.803 \end{pmatrix} = \begin{pmatrix} 19.724 \\ 8.164 \end{pmatrix}.$$

The unit vector along this direction is $(0.924,\ 0.382)^T$. It is interesting to observe that this search direction is similar to the first search direction $s^{(1)}$ obtained by using the Fletcher-Reeves algorithm.

Step 4: Performing a unidirectional search along $s^{(1)}$ from the point $x^{(1)}$, we obtain the minimum point: $x^{(2)} = (2.248, 2.989)^T$ having a function value $f(x^{(2)}) = 26.237$ (Figure 3.2).

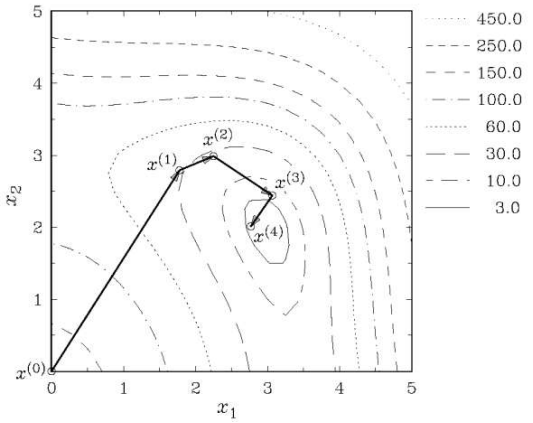

Figure 3.2: Three iterations of the DFP method.

Step 5: We observe that the point $x^{(2)}$ is very different from the point $x^{(1)}$ and calculate the gradient at $x^{(2)}$: $\nabla f(x^{(2)}) = (-18.225, 44.037)^T$. Since $\left\| \nabla f(x^{(2)}) \right\|$ is not close to zero, we increment *k* and proceed to Step 4. This completes one iteration of the DFP method. The initial point $x^{(0)}$, the point $x^{(1)}$, and the point $x^{(2)}$ are shown in Figure 3.2.

Step 4: The second iteration begins with computing another search direction $s^{(2)} = -A^{(2)}\nabla f(x^{(2)})$. The matrix $A^{(2)}$ can be computed by calculating $\Delta x^{(1)}$ and $\Delta e(x^{(1)})$ as before and using Equation (3.2). By simplifying the expression, we obtain the new matrix

$$A^{(2)} = \begin{pmatrix} 0.070 & -0.017 \\ -0.017 & 0.015 \end{pmatrix}$$

and the new search direction is found to be

$$s^{(2)} = (0.731, -0.976)^T$$

Note that this direction is more descent than that obtained in the Fletcher-Reeves method after one iteration.

Step 5: The unidirectional search along $s^{(2)}$ finds the best point: $x^{(3)} = (2.995, 1.991)^T$ with a function value equal to $f(x^{(3)}) = 0.003$.
Another iteration updates the A matrix as follows:

$$A^{(3)} = \begin{pmatrix} 0.030 & -0.005 \\ -0.005 & 0.020 \end{pmatrix}.$$

The new search direction is found to be $s^{(3)} = (0.014, 0.095)^T$. The unidirectional search method finds the new point $x^{(4)} = (2.997, 2.000)^T$ with a function value equal to 3×10^4. Another iteration finds the following A matrix:

$$A^{(4)} = \begin{pmatrix} 0.018 & -0.011 \\ -0.011 & 0.036 \end{pmatrix}$$

and the new search direction $s^{(4)} = (0.003, -0.003)^T$. A unidirectional search finds the point $x^{(5)} = (3.000, 2.000)^T$ with a function value equal to zero. One final iteration finds the A matrix:

$$A^{(5)} = \begin{pmatrix} 0.016 & -0.009 \\ -0.009 & 0.035 \end{pmatrix}.$$

which is identical to the inverse of the Hessian matrix of the Himmelblau function at the minimum point (Equation (3.3)). The above calculation shows how the original identity matrix $A^{(0)}$ has transformed into $H^{-1}(x^*)$ matrix in successive iterations. From the fourth iteration onwards, the search direction is very similar to that in the Newton's method, because the matrix $A^{(k)}$ is similar to the inverse of the Hessian matrix. Since, during that iteration the solution is also close to the minimum, the search converges faster to the minimum point. One difficulty with this method is that the matrix A sometimes becomes ill-conditioned due to numerical derivative computations and due to inaccuracy involved in unidirectional searches.

3.4. References

[3.1] Box, G. E. P. and Draper, N. R. (1969): **Evolutionary Operation.** New York: Wiley.

[3.2] Davidon, W. C. (1959): **Variable metric method for minimization** (Technical Report No. ANL-599). AEC Research Development Report.

[3.3] Fletcher, R. and Powell, M. J. D. (1963): **A rapidly convergent descent method for minimization.** Computer Journal. 6. 163-168.

[3.4] Fletcher, R. and Reeves, C. M. (1964): **Function minimization by conjugate gradients.** Computer Journal. 7, 149-154.

[3.5] Kreyszig, E. (1983): **Advanced Engineering Mathematics.** New Delhi: Wiley Eastern.

[3.6] Neider, J. A. and Mead, R. (1965): **A simplex method for function minimization.** Computer Journal. 7, 308-313.

[3.7] Rao, S. S. (1984): **Optimization Theory and Applications.** New Delhi: Wiley Eastern.

[3.8] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983): **Engineering Optimization-Methods and Applications.** New York: Wiley.

3.5. Questions

1. Explain the sufficient and the necessary conditions of the optimality!
2. What properties fulfill conjugate directions in Powell's method? Why?
3. What compromise is reached by Marquardt's Method? How?
4. How is the Hessian matrix used in DFP method?

4. CONSTRAINED OPTIMIZATION

Single and multi-variable functions with equality or inequality constraints or both are considered.

4.1. Kuhn-Tucker Conditions

We assume that objective function $f(x)$ (x is an N-dimensional array), inequality constraints $g_j(x), j = 1, 2, ..., J$, and equality constraints $h_k(x), k = 1, 2, ..., K$ are all differentiable. The NLP problem is as follows:

$$\begin{array}{lll} \text{Minimize} & f(x) & \\ \text{Subject to} & g_j(x) \geq 0 & j = 1,2 \ldots, J \\ & h_k(x) = 0 & k = 1,2 \ldots, K \end{array}$$

The Kuhn-Tucker problem is to find vectors x (size N), u (size J), and v (size K) that satisfy

$$\nabla f(x) - \sum_{j=1}^{J} u_j \nabla g_j(x) - \sum_{k=1}^{K} \nabla h_k(x) = 0 \quad N \text{ equations}$$

$$\begin{array}{rl} g_j(x) \geq 0 & j = 1,2, \ldots, J \\ h_k(x) = 0 & k = 1,2, \ldots, K \\ u_j g_j(x) = 0 & j = 1,2, \ldots, J \\ u_j \geq 0 & j = 1,2, \ldots, J \end{array}$$

If $g_j(\bar{x}) = 0$, the constraint j is active or binding at $\bar{x}$.
If $g_j(\bar{x}) > 0$, the constraint j is inactive or nonbinding at $\bar{x}$.

Kuhn-Tucker Necessity Theorem:
Consider the NLP problem shown above. Let f, g, and h be differentiable functions and x^* be a feasible solution to NLP. Let $I = \{j \mid g_j(x^*) = 0\}$. Furthermore, $\nabla g_j(x^*)$ for $j \in I$ and $\nabla h_k(x^*)$ for $k = 1, 2, \ldots, K$ are linearly independent (Constraint qualification). If x^* is an optimal solution to NLP, there exists a (u^*, v^*) such that (x^*, u^*, v^*) solves Kuhn-Tucker problem.
If a feasible point does not satisfy constraint qualification, K-T necessity theorem can be used to prove that the point is not optimal but not vice versa.

Kuhn-Tucker Sufficiency Theorem:
Let the objective function be convex, the inequality constraints $g_j(x)$ be all concave functions for $j = 1, 2, \ldots, J$ and equality constraints $h_k(x)$ for $k = 1, 2, \ldots, K$ be linear. If there exists a solution (x^*, u^*, v^*) that satisfies the K-T conditions, then x^* is an optimal solution to the NLP problem.

4.2. Transformation Methods

- Original constrained problem is transformed into a sequence of unconstrained problems via penalty functions
- Methods vary according to the way constraints are handled.

Interior penalty method: Each sequence contains feasible points
Exterior penalty method: Each sequence contains infeasible points
Mixed penalty method: Each sequence contains feasible or infeasible points

4.2.1. Penalty Function Method

- Penalty concept:

$$P(x, R) = f(x) + \Omega(R, g(x), h(x))$$

where R is a set of penalty parameters, Ω is the penalty term so selected to favor the selection of feasible points over infeasible points:

1. Parabolic Penalty: $\Omega = R\{h(x)\}^2$. Used for equality constraints and it is an exterior penalty term. A small value of R is started with and increased gradually.
2. Infinite Barrier Penalty: $\Omega = 10^{20}\sum_{j\in\bar{j}}\left|g_j(x)\right|$. It assigns an infinite penalty to infeasible points.
3. Log Penalty: $\Omega = -R\ln[g(x)]$. A large value of R is started with and reduced to zero gradually. It is an interior penalty term.
4. Inverse Penalty: $\Omega = R\left[\frac{1}{g(x)}\right]$. Like Log penalty, the value of R starts from large value and reduces to zero. It is also an interior penalty term.
5. Bracket Operator Penalty: $\Omega = R\langle g(x)\rangle^2$, where $\langle\alpha\rangle = \alpha$, when a is negative; zero, otherwise. Here R starts from a small value and increases to a large value. This is an exterior penalty term.

Algorithm:

Step 1: Choose two termination parameters $\varepsilon_1, \varepsilon_2$, an initial solution $x^{(0)}$, a penalty term Ω, and an initial penalty parameter $R^{(0)}$. Choose a parameter c to update R such that $0 < c < 1$ is used for interior penalty terms and $c > 1$ is used for exterior penalty terms. Set $t = 0$.

Step 2: Form $P(x^{(t)}, r^{(t)}) = f(x^{(t)}) + \Omega(R^{(t)}, g(x^{(t)}), h(x^{(t)}))$

Step 3: Starting with a solution $x^{(t)}$, find $x^{(i+1)}$ such that $P(x^{(i+1)},R^{(t)})$ is minimum for a fixed value of $R^{(t)}$. Use ε_1 to terminate the unconstrained search.

Step 4: Is $\left|P(x^{(t)}, R^{(t)}) - P(x^{(t)}, R^{(t-1)}\right| \le \varepsilon_2$?

If yes, set $x^T = x^{(t+1)}$ and **Terminate.**
Else go to Step 5.

Step 5: Choose $R^{(t+1)} = cR^{(t)}$. Set $t = t + 1$ and go to Step 2.

Consider the constrained Himmelblau's function:

$$\text{Minimize} \qquad (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

subject to

$$(x_1 - 5)^2 + x_2^2 - 26 \geq 0. \qquad x_1, x_2 \geq 0.$$

The inclusion of the constraint changes the unconstrained optimum point. The feasible region and the optimum point of this NLP is shown in Figure 4.1. The figure shows that the original optimum point $(3,1)^T$ is now an infeasible point. The new optimum is a point on the constraint line that touches a contour line at that point.

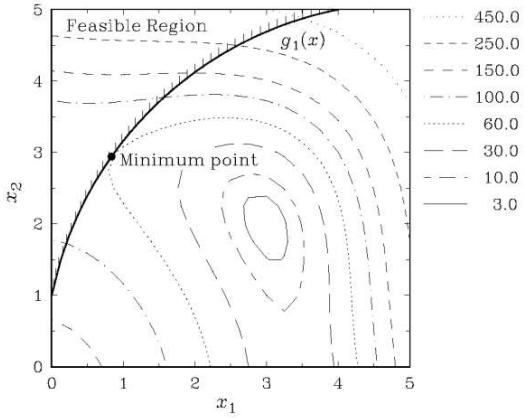

Figure 4.1: The feasible search space and the true minimum of the constrained problem.

Step 1: We use the bracket-operator penalty term to solve this problem. The bracket operator penalty term is an exterior penalty term. We choose an infeasible point $x^{(0)} = (0,0)^T$ as the initial point. We also choose a small value for the penalty parameter: $R^{(0)} = 0{,}1$. We choose two convergence parameters $\varepsilon_1 = \varepsilon_2 = 10^{-5}$.

Step 2: The next task is to form the penalty function:

$$P(x, R^{(0)}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 0.1 \times \langle (x_1 - 5)^2 + x_2^2 - 26 \rangle^2$$

In the above formulation, the variable bounds must also be included as inequality constraints. For clarity and ease of illustration, we have not included the variable bounds in our formulation.

Step 3: The above unconstrained function can now be minimized using one of the methods described in the previous chapter. Here, we use the steepest descent method to solve the above problem. We begin the algorithm with an initial solution $x^{(0)} = (0,0)^T$ having $f(x^{(0)}) = 170{,}0$. At this point, the constraint violation is -1.0 and the penalized function value $P(x^{(0)}, R^{(0)}) = 170{,}100$. Intermediate points obtained by the steepest descent algorithm are tabulated in Table 4.1, and some of these points are shown in Figure 4.2. After

150 function evaluations, the solution $x^* = (2.628, 2.475)^T$ having a function value equal to $f(x^*) = 5.709$ is obtained. At this point, the constraint violation is equal to (-14.248), but has a penalized function value equal to 25.996, which is smaller than that at the initial point. Even though the constraint violation at this point is greater than that at the initial point, the steepest descent method has minimized the penalized function $P(x, R^{(0)})$ from 170.100 to 25.996. We set $x^{(1)} = (2.628, 2.475)^T$ and proceed to the next step.

Sequence t	$R^{(t)}$	Solution $x^{(t)}$	$P(x^{(t)}, R^{(t)})$	Constraint violation
1	0.1	$(0, 0)^T$	170.100	−1.000
		$(2.569, 2.294)^T$	27.119	−14.828
		$(2.657, 2.455)^T$	26.019	−14.483
		.	.	.
		$(2.628, 2.475)^T$	25.996	−14.248
2	1.0	$(2.628, 2.475)^T$	208.70	−14.248
		$(1.730, 3.412)^T$	75.140	−3.655
		$(1.166, 2.871)^T$	60.986	−3.058
		.	.	.
		$(1.011, 2.939)^T$	58.757	−1.450
3	10.0	$(1.011, 2.939)^T$	77.591	−1.450
		$(0.906, 3.016)^T$	60.530	−0.143
		.	.	.
		$(0.844, 2.934)^T$	60.233	−0.119

Table 4.1: Tabulation of the Intermediate Points Obtained Using the Steepest Descent Algorithm

Step 4: Since this is the first iteration, we have no previous penalized function value to compare with; thus we move to Step 5.

Step 5: At this step, we update the penalty parameter $R^{(1)} = 10 \times 0{,}1 = 1{,}0$ and move to Step 2. This is the end of the first sequence. It is important here to note that with a different initial point, we could have also converged to the same point. But simulation runs with certain initial points may have taken a longer time to converge than with other points. However, solutions in subsequent sequences will be identical for all simulations.

Step 2: The new penalized function in the second sequence is as follows:

$$P(x, R^{(1)}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 1.0 \times \{(x_1 - 5)^2 + x_2^2 - 26\}^2.$$

Step 3: At this step, we once again use the steepest descent method to solve the above problem from the starting point $(2.628, 2.475)^T$. Table 4.1 shows intermediate points of the simulation run. The minimum of the function is found after 340 function evaluations and is $x^{(2)} = (1.011, 2.939)^T$. At this point, the constraint violation is equal to -1.450, which suggests that the point is still an infeasible point. The penalized function and the minimum of the function are both shown in Figure 4.3. The progress of the previous sequence is also shown using dashed lines. Observe that this penalized function is distorted with respect to the original Himmelblau function. This distortion is necessary to shift the minimum point of the current function closer to the true constrained minimum point. Also notice that the penalized function at the feasible region is undistorted.

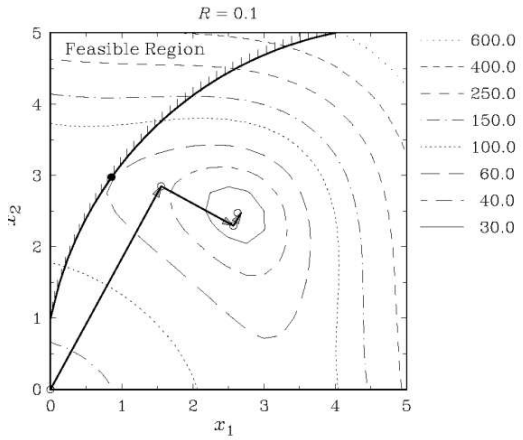

Figure 4.2: A simulation of the steepest descent method on the penalized function with R = 0.1.

Step 4: Comparing the penalized function values, we observe that $P(x^{(2)}, 1, 0) = 58.664$ and $P(x^{(1)}, 1, 0) = 25.996$. Since they are very different from each other, we continue with Step 5.

Step 5: The new value of the penalty parameter is $R^{(2)} = 10,0$. We increment the iteration counter $t = 2$ and go to Step 2.

In the next sequence, the penalized function is formed with $R^{(2)} = 10,0$. The penalized function and the corresponding solution is shown in Figure 4.4. This time the steepest descent algorithm starts with an initial solution $x^{(2)}$. The minimum point of the sequence is found to be $x^{(3)} = (0.844, 2.934)^T$ with a constraint violation equal to -0.119. Figure 4.4 shows the extent of distortion of the original objective function. Compare the contour levels shown at the top right corner of Figures 4.2 and 4.4. With $R = 10.0$, the effect of the objective function $f(x)$ is almost insignificant compared to that of the constraint violation in the infeasible search region. Thus, the contour lines are almost parallel to the constraint line. Fortunately in this problem, the increase in the penalty parameter R only makes the penalty function steeper in the infeasible search region. In problems with a sufficiently nonlinear objective function and with multiple constraints, a large value of the penalty parameter may create one or more artificial local optima in the search space, thereby making it difficult for the unconstrained search to obtain the correct solution. The advantage of using the unconstrained search method sequentially is that the unconstrained search is always started from the best point found in the

previous sequence. Thus, despite the presence of many local optima in the search space, the search at every sequence is initiated from a point near the correct optimum point. This makes it easier for the unconstrained search to find the correct solution.
After another sequence (iteration) of this algorithm, the obtained solution is

$$x^{(4)} = (0.836, 2.940)^T$$

with a constraint violation of only - 0.012. This point is very close to the true constrained optimum solution. A few more iterations of the penalty function method may be performed to get a solution with the desired accuracy. Although a convergence check with a small difference in the penalized function value at two consecutive sequences is used in this algorithm, any

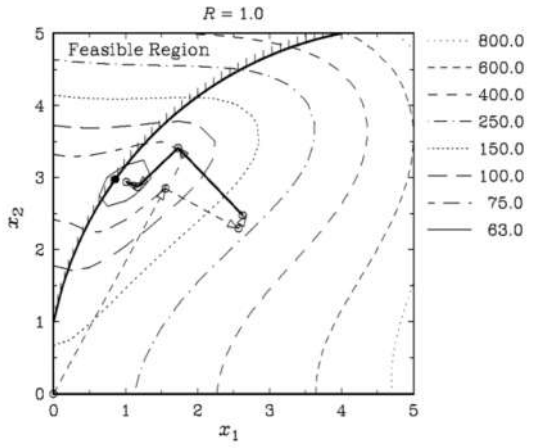

Figure 4.3: Intermediate points using the steepest descent method for the penalized function with R=1.0 (solid lines).

other convergence criteria (for example, a small difference in the *x*-vector of the solutions in two successive sequences) may also be used.
In the presence of multiple constraints, it is observed that the performance of the penalty function method improves considerably if the constraints and the objective functions are first normalized before constructing the penalized function. An inequality constraint $g_j(x) \geq 0$ can be normalized as follows:

$$\frac{g_j(x)}{g_{\max}} \geq 0,$$

where g_{max} is the maximum value of the constraint $g_j(x)$ in the search space. Often, engineering design problems contain constraints restraining resource or capacity of b_j as $g'_j(x) \leq b_j$. The constraint can be normalized as follows:

$$1 - \frac{g'_j(x)}{b_j} \geq 0$$

If an upper bound of the objective function is known, the objective function can also be normalized as shown above, but the normalization of the constraints is more important than that of the objective function.

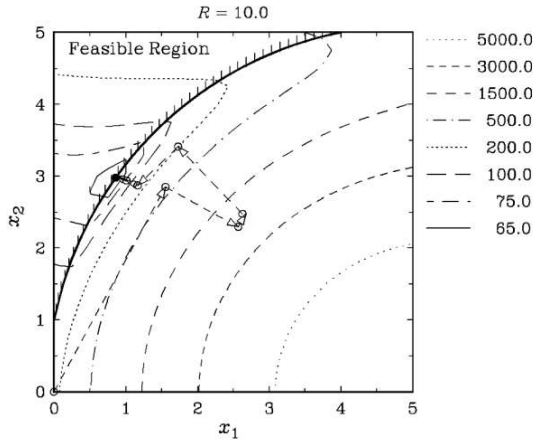

Figure 4.4: Intermediate points obtained using the steepest descent method for the penalized function with R = 10.0 (solid lines near the true optimum). Notice the distortion in the function.

4.2.2. Method of Multipliers

- Distortion of functions is avoided by using fixed parameter penalty method
- The following penalty function and update rules are used:

$$P\left(x,\sigma^{(t)},\tau^{(t)}\right) = f(x) + R\sum_{j1}^{J}\{(\{g_j(x)+\sigma_j^{(t)}\})^2 - (\sigma_j^{(t)})^2\}$$

$$+R\sum_{k=1}^{K}\{(h_k(x)+\tau_k^{(t)})^2 - (\tau_k^{(t)})^2\}.$$

The final solution x^T is a K-T point. Furthermore, the Lagrange multipliers are computed easily:

$$u_j = -2R\sigma_j^T$$

$$v_k = -2R\tau_k^T$$

Algorithm:

Step 1: Choose a penalty parameter R, termination parameters ε_1 and ε_2. Choose an initial solution $x^{(0)}$. Set multipliers $\sigma_j^{(0)} = \tau_k^{(0)}$ and the iteration counter $t = 0$.

Step 2: Next, form the penalized function:

$$P(x,\sigma^{(t)},\tau^{(t)}) = f(x) + R\sum_{j1}^{J}\{(\{g_j(x)+\sigma_j^{(t)}\})^2 - (\sigma_j^{(t)})^2\}$$

$$+R\sum_{k1}^{K}\{(h_k(x)+\tau_k^{(t)})^2 - (\tau_k^{(t)})^2\}.$$

Step 3: Use an unconstrained optimization technique to solve the penalized function $P(x,\sigma^{(t)},\tau^{(t)})$ from the starting point $x^{(t)}$ with an convergence factor ε_1. During

this optimization, $\sigma^{(t)}$ and $\tau^{(t)}$ are kept fixed; only the vector x is varied. Let us say the solution is $x^{(t+1)}$.

Step 4: Is $\left\| P(x^{(t+1)}, \sigma^{(t)}, \tau^{(t)}) - P(x^{(t)}, \sigma^{(t-1)}, \tau^{(t-1)}) \right\| \le \varepsilon_2$?

If yes, set $x^T = x^{(t+1)}$ and **Terminate.**
Else go to Step 5.

Step 5: Update $\sigma_j^{(t+1)} = \left\langle g_j(x^{(t)}) + \sigma_j^{(t)} \right\rangle$ for all $j = 1,2,\ldots, J$ and $\tau_k^{(t+1)} = h_k(x^{(t)}) + \tau_k^{(t)}$ for all $k = 1,2,\ldots, K$. Set $t = t+1$ and go to Step 2.

4.3. Constrained Direct Search

- Detailed structure of constraints are considered,
- Functions may be discontinuous and non-differentiable,
- Algorithms are heuristic in nature,
- Algorithms start from a feasible point. When a new point is created by a fixed rule make sure that the point is feasible. If not, modify by a predefined rule.

4.3.1. Random Search Methods

Points are generated at random.

Luus and Jaakola Algorithm:

Step 1: Given an initial feasible point x^0, an initial range z^0 such that the minimum, x^*, lies in $(x^0 - \frac{1}{2}z^0, x^0 + \frac{1}{2}z^0)$. Choose a parameter $0 < \varepsilon < 1$. For each of Q blocks, initially set $q = 1$ and $p = 1$.

Step 2: For $i = 1,2,\ldots, N$, create points using a uniform distribution of r in the range (-0.5, 0.5). Set $x_i^{(p)} = x_i^{q-1} + rz_i^{q-1}$.

Step 3: If $x^{(p)}$ is infeasible and $p < P$, repeat Step 2.

If $x^{(p)}$ is feasible, save $x^{(p)}$ and $f(x^{(p)})$, increment p and repeat Step 2.

Else if $p = P$, set x^q to be the point that has the lowest $f(x^{(p)})$ over all feasible $x^{(p)}$ including x^{q-1} and reset $p = 1$

Step 4: Reduce the range via $z_i^q = (1-\varepsilon) z_i^{q-1}$

Step 5: If $q > Q$, **Terminate.**

Else increment q and continue with Step 2

Suggested values of parameters are $\varepsilon = 0.05$, $p = 100$, and Q is related to the desired reduction in variable uncertainity.

4.4. Method of Feasible Directions

- Rather than relying on solutions to LP problems, use linear approximations to determine a locally good search direction,
- Similar to gradient-based unconstrained search,

– Zoutendijk's method: Have a search direction d that is descent $(\nabla f(x^{(t)}) \cdot d < 0)$ and feasible $(\nabla g(x^{(t)}) \cdot d \geq 0)$.

Algorithm:

Step 1: Set an iteration counter $t = 0$. Choose an initial feasible point $x^{(0)}$ and a parameter for checking the constraint violation, ε.

Step 2: At the current point $x^{(t)}$, let $I^{(t)}$ be the set of indices of active constraints. In other words,

$$I^{(t)} = \{j: 0 \leq g_j(x^{(t)}) \leq \varepsilon, j = 1,2,..)J\}.$$

If $I^{(t)}$ is empty, use $d^{(t)} = \theta^{(t)} = -\nabla f(x^{(t)})$, normalize $d^{(t)}$ and go to Step 4

Step 3: Solve the following LP:

Maximize θ
subject to

$$\nabla f(x^{(t)})d \leq -\theta,$$
$$\nabla g_j(x^{(t)})d \geq \theta, \quad j \in l^{(t)},$$
$$-1 \leq l \leq 1, \qquad i = 1,2, ..., N.$$

Label the solution $d^{(t)}$ and $\theta^{(t)}$.

Step 4: If $Q(t) < 0$, **Terminate.**

Else $\bar{\alpha} = \min\left[\alpha : g_j(x^{(t)} + \alpha d^{(t)}) = 0,\ j = 1, 2, ..., J, \text{ and } \alpha > 0\right]$.

If no $\bar{\alpha} > 0$ exists, set $\bar{\alpha} = \infty$.

Step 5: Find $\alpha^{(t)}$ such that

$$f(x^{(t)} + \alpha^{(t)} d^{(t)}) = \min\left[f(x^{(t)} + \alpha d^{(t)}) : 0 \leq \alpha \leq \overline{\alpha}\right]$$

Set $x^{(t+1)} = x^{(t)} + \alpha^{(t)} d^{(t)}$, $t = t + 1$, and go to Step 2

Comments:

1. Only a subset of constraints are used to define a subproblem, thus LP is smaller,
2. Since only binding constraints are used, zigzag iteration pattern results.

Consider the constrained Himmelblau function again:

$$\text{Minimize } f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

$$g_1(x) = 26 - (x_1 - 5)^2 - x_2^2 \geq 0,$$
$$g_2(x) = 20 - 4x_1 - x_2 \geq 0,$$
$$x_1, x_2 \geq 0.$$

Let us recall that the minimum point of the above problem lies at $x^* = (3, 2)^T$ with a function value equal to zero.

Step 1: Let us choose an initial feasible point $x^{(0)} = (0,0)^T$ and a tolerance parameter $\varepsilon = 10^{-3}$. We also set the iteration counter $t = 0$.

Step 2: At this step, let us find the active constraints at point $x^{(0)}$. It turns out that only variable bounds are active. Calling these two inequality constraints $g_3(x) = x_1 \geq 0$ and $g_4(x) = x_2 \geq 0$, we update the active constraint set $I^{(0)} = \{3, 4\}$. Since this set is not empty, we continue with Step 3.

Step 3: At this step, we have to find a descent direction which is maximally away from both constraints g_3 and g_4. At the initial point, both active constraints are orthogonal to each other. Thus, the desired search direction may make equal angle with all active constraints. But in any other situation, this may not be true. Thus, we find the optimal search direction by solving an LP problem. Calculating the derivative of the objective function and the active constraints at $x^{(0)}$ numerically and denoting the search direction $d = (d_1, d_2)^T$, we obtain the following LP problem:

Maximize θ
subject to

$$-14d_1 - 22d_2 \leq -\theta$$
$$d_1 \geq \theta,$$
$$d_2 \geq \theta,$$
$$-1 \leq d_{1,} d_2 \leq 1.$$

There exist a number of difficulties with the above formulation to be directly solved using the simplex method of LP technique. First of all, in the above formulation, the variables can take negative values, which are not allowed in a LP technique. Thus, we first substitute $t_i = d_i + 1$ for $i = 1, 2$ such that the variables t_i can take only positive values in the range (0, 2). We rewrite the above LP problem in terms of the new variables:

Maximize θ
subject to

$$14t_1 + 22t_2 - \theta \geq 36$$
$$t_1 - \theta \geq 1,$$
$$t_2 - \theta \geq 1,$$
$$0 \leq t_{1)} t_2 \leq 2.$$

Secondly, the simplex method can handle only equality constraints. Slack variables are usually added or subtracted to convert inequality constraints to equality constraints. Therefore, for each of the above constraints we add a slack variable (y_1 to y_5). Thirdly, we observe that the problem variables and the slack variables do not constitute an initial basic feasible solution. Thus, we introduce three more artificial variables y_6, y_7, and y_8) to constitute an initial basic feasible solution for the first phase of the dual simplex search method. Thus, the underlying LP problem becomes as follows:

Maximize θ
subject to

$$14t_1 + 22t_2 - \theta - y_1 + y_6 = 36,$$

$$\begin{gathered} t_1 - \theta - y_2 + y_7 = 1, \\ t_2 - \theta - y_3 + y_8 = 1, \\ t_1 + y_4 = 2, \\ t_2 + y_5 = 2, \\ t_1, t_2, y_1, y_2, t_3, y_4, y_5, y_6, y_7, y_8 \geq 0. \end{gathered} \tag{4.1}$$

The three-variable problem now becomes an 11-variable problem. At first, we solve the above problem for the objective:

Maximize θ
subject to

$$-(y_6 + y_7 + y_8).$$

Since all artificial variables must also be nonnegative, the solution to the above problem would have $y_6 = y_7 = y_8 = 0$, because the above objective function at this point would be zero. This solution will then be a candidate solution for the initial basic feasible solution of the problem

		0	0	0	0	0	0	0	0	−1	−1	−1		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8		
−1	y_6	14	22	−1	−1	0	0	0	0	1	0	0	36	$\frac{36}{22} = 1.64$
−1	y_7	1	0	−1	0	−1	0	0	0	0	1	0	1	$\frac{1}{0} = \infty$
−1	y_8	0	**1**	−1	0	0	−1	0	0	0	0	1	1	$\frac{1}{1} = 1$ ←
0	y_4	1	0	0	0	0	0	1	0	0	0	0	2	$\frac{2}{0} = \infty$
0	y_5	0	1	0	0	0	0	0	1	0	0	0	2	$\frac{2}{1} = 2$
$(\Delta f)_q$		15	23	−3	−1	−1	−1	0	0	0	0	0	$f = -38$	
			↑											

Table 4.2: The First Tableau for the First Phase of the Dual Simplex Search Method

		0	0	0	0	0	0	0	0	−1	−1	−1		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8		
−1	y_6	14	0	21	−1	0	**22**	0	0	1	0	−22	14	0.64 ←
−1	y_7	1	0	−1	0	−1	0	0	0	0	1	0	1	$\frac{1}{0} = \infty$
0	t_2	0	1	−1	0	0	−1	0	0	0	0	1	1	−ve
0	y_4	1	0	0	0	0	0	1	0	0	0	0	2	$\frac{2}{0} = \infty$
0	y_5	0	0	1	0	0	1	0	1	0	0	−1	1	$\frac{1}{1} = 1$
$(\Delta f)_q$		15	0	20	−1	−1	22	0	0	0	0	−23	$f = -15$	
						↑								

Table 4.3: The Second Tableau for the First Phase of the Dual Simplex Search Method

presented in Equation (4.1). The successive tables for the first problem of obtaining a feasible starting solution are shown in Tables 4.5 to 4.8.

The objective function value at the Table 4.5 is $f = -38$. (Equation (4.2) is used as the objective.) In the table, it is clear that the nonbasic variable t_2 corresponds to a maximum increase in the function value. (The quantity $(\Delta f)_q$ is larger for t_2.) Thus, we choose t_2 as the new basic variable.

It turns out from the minimum ratio rule that the basic variable y_8 must be replaced by the the variable t_2 in the next iteration. We formulate the next row-echelon matrix. The outcome of the calculation is shown in Table 4.6.

Note that the objective function value has improved considerably from the previous iteration. Here, we also observe that the nonbasic variable y_3 corresponds to the maximum value of the quantity $(\Delta f)_q$. Thus, we choose y_3 as the new basic variable. Using the minimum ratio rule, we also observe that the current basic variable y_6 must be replaced by the variable y_3 (Table 4.7). At the end of the third iteration, we observe that the nonbasic variable t_1 must replace the basic variable y_7. The objective function value at this iteration is $f = -1$.

		0 0	0	0	0 0 0 0	0	−1 −1	−1	
c_B	Basic	t_1 t_2	θ	y_1	y_2 y_3 y_4 y_5	y_6	y_7	y_8	
0	y_3	0.64 0	0.95	−0.04	0 1 0 0	0.04	0	−1	0.64
−1	y_7	**1.00** 0	−1.00	0.00	−1 0 0 0	0.00	1	0	1.00 ←
0	t_2	0.64 1	−0.04	−0.04	0 0 0 0	0.04	0	0	1.64
0	y_4	1.00 0	0.00	0.00	0 0 1 0	0.00	0	0	2.00
0	y_5	−0.64 0	0.04	0.04	0 0 0 1	−0.04	0	0	0.36
$(\Delta f)_q$		1.00 0	−1.00	0.00	−1 0 0 0	−1.00	0	−1	$f = -1$
		↑							

Ratios: 1 (first row), 1 (second row), 2.57 (third row), 2 (fourth row), and −ve (fifth row)

Table 4.4: The Third Tableau for the First Phase of the Dual Simplex Search Method

We form the next row-echelon matrix in Table 4.8. At this stage, we observe that all artificial

		0 0	0	0	0 0 0 0	−1	−1	−1	
c_B	Basic	t_1 t_2	θ	y_1	y_2 y_3 y_4 y_5	y_6	y_7	y_8	
0	y_3	0 0	1.59	−0.04	0.64 1 0 0	0.04	−0.64	−1	0
0	t_1	1 0	−1.00	0.00	−1.00 0 0 0	0.00	1.00	0	1
0	t_2	0 1	0.59	−0.04	0.64 0 0 0	0.04	−0.64	0	1
0	y_4	0 0	1.00	0.00	1.00 0 1 0	0.00	−1.00	0	1
0	y_5	0 0	−0.59	0.04	0.04 0 0 1	−0.04	0.64	0	1
$(\Delta f)_q$		0 0	0.00	0.00	0 0 0 0	−1.00	−1.00	−1	$f = 0$

Table 4.5: The Fourth Tableau for the First Phase of the Dual Simplex Search Method

variables are zero and the objective function is also equal to zero. This is the termination criterion for the first phase of the dual phase method. The solution of the above iteration is $t_1 = 1,\ t_2 = 1,\ y_1 = 0,\ y_2 = 0,\ y_3 = 0,\ y_4 = 1$, and $y_5 = 1$. This solution was not obvious in the formulation of the problem presented in Equation (4.1).
We begin the second phase with the above solution as the initial solution. The objective in the second phase is to maximize the original function: $f(x) = \theta$. Since the artificial variables are no more required, we discontinue with them in subsequent computations.

		0 0	1	0	0 0 0 0				
c_B	Basic	t_1 t_2	θ	y_1	y_2 y_3 y_4 y_5				
0	y_3	0 0	1.59	−0.04	0.64 1 0 0	0	$\frac{0}{1.59} = 0$	←
0	t_1	1 0	−1.00	0.00	−1.00 0 0 0	1	−ve	
0	t_2	0 1	0.59	−0.04	0.64 0 0 0	1	$\frac{1}{0.59} = 1.69$	
0	y_4	0 0	1.00	0.00	1.00 0 1 0	1	$\frac{1}{1} = 1$	
0	y_5	0 0	−0.59	0.04	0.04 0 0 1	1	−ve	
$(\Delta f)_q$		0 0	1.00	0.00	0.00 0 0 0	$f(x) = 0$		
			↑					

Table 4.6: The First Tableau for the Second Phase of the Dual Simplex Search Method

Now, we get back to Step 3 of the feasible direction search method.

Step 3: **(cont.)** The solution of the LP problem is $t = (2, 2)^T$ or $d^{(0)} = (1, 1)^T$ and $\theta^{(0)} = 1$. This solution implies that the resulting search direction makes equal angles with each of the two active constraints.

Step 4: Since $\theta^{(0)} = 1 > 0$, we do not terminate the algorithm. Instead, we calculate the limits along the direction $d^{(0)}$ before an infeasible point is found. Any generic point along $d^{(0)}$ from $x^{(0)}$ can be written as $x(\alpha) = x^{(0)} + \alpha d^{(0)}$ or $x(\alpha) = (\alpha, \alpha)^T$. The upper limit on a can be calculated by finding points along $d^{(0)}$ that intersect with each constraint. The problem of finding the intersection of a straight line and any generic curve can be posed as a root-finding problem, which can be solved using an optimization algorithm discussed in Chapter 2. We substitute the expression for $x_1 = \alpha$ and $x_2 = \alpha$ in each constraint and then minimize the following problem:

$$\text{Minimize abs } [g_j(x(\alpha))]. \tag{4.3}$$

For example, the upper limit along $d^{(0)}$ can be found for the first constraint by minimizing the unidirectional function:

$$\text{abs } [26 - (\alpha - 5)^2 - \alpha^2]$$

Since the absolute value of the argument is always considered, the above function allows only positive values. Since we are looking for points for which the constraint has a value zero, those points correspond to the minimum value of the above expression. Note that the problem

described in Equation (4.3) is a single-variable function. Thus, we first bracket the minimum and then minimize the function. Using the bounding phase method from a starting point $\alpha^{(0)} = 5$ and $\Delta = 1$, we obtain the bracketing interval (4, 6). Next, we use the golden section search in that interval to obtain the minimum point with three decimal places of accuracy: $\alpha = 5.098$. The same solution can also be obtained by solving the quadratic expression $g_1(x(\alpha)) = 0$. Similarly, the limit on the second constraint can also be calculated: $\alpha_2^* = 4{,}0$. Other constraints produce upper limits $\alpha_3^* = \alpha_4^* = 0$, which are not acceptable. Thus, the true upper limit is $\alpha = 4{,}0$.

		0	0	1	0	0	0	0	0		
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5		
1	θ	0	0	1	−0.03	0.40	0.63	0	0	0	−ve
0	t_1	1	0	0	−0.03	−0.60	0.63	0	0	1	$\frac{1}{0} = \infty$
0	t_2	0	1	0	−0.03	0.40	−0.37	0	0	1	−ve
0	y_4	0	0	0	0.03	0.60	0.63	1	0	1	$\frac{1}{0.03} = 35$ ←
0	y_5	0	0	0	0.03	−0.40	0.37	0	1	1	$\frac{1}{0.03} = 35$
$(\Delta f)_q$		0	0	0	0.03	−0.4	−0.63	0	0	$f(x) = 0$	

↑

Table 4.7: The Second Tableau for the Second Phase of the Dual Simplex Search Method

		0	0	1	0	0	0	0	0	
c_B	Basic	t_1	t_2	θ	y_1	y_2	y_3	y_4	y_5	
1	θ	0	0	1	0	1	0	1	0	1
0	t_1	0	0	0	0	0	0	1	0	2
0	t_2	0	1	0	0	1	−1	1	0	2
0	y_1	0	0	0	1	21	−22	1	0	35
0	y_5	0	0	0	0	−1	1	−1	1	0
$(\Delta f)_q$		0	0	0	0	−1	0	−1	0	$f(x) = 1$

Table 4.8: The Third Tableau for the Second Phase of the Dual Simplex Search Method

Step 5: Once the lower and upper limit on a are found, we perform another one dimensional search with the given objective function to find the minimum point along that direction. Using the golden section search, we obtain the minimum point in the interval (0, 4): $\alpha^* = 2.541$, which corresponds to the new point $x^{(1)} = (2.541, 2.541)^T$. At this point, we increment the iteration counter and go to Step 2. This completes one iteration of the feasible direction method. The progress of this iteration is shown in Figure 4.5.

Step 2: At the new point, we find that no constraints are active. Thus, $I^{(1)} = \emptyset$, which means that the point is not on any constraint boundary and we are free to search in any

direction locally. Therefore, we choose the steepest descent direction and the search direction is set according to the negative of the gradient of the objective function at the new point:

$$d^{(1)} = -\nabla f(x^{(1)}) = (16.323, -16.339)^T,$$

which is computed numerically. At this point the function value is $f(x^{(1)}) = 8{,}0$.

Step 4: Once again, we compute the upper limit along the search direction $d^{(1)}$. Posing the root-finding problem as an optimization problem as shown in the previous iteration, we obtain the parameter $\overline{\alpha} = \min[0.162, 0.149] = 0.149$.

Step 5: Performing a unidirectional search along in the domain (0, 0.149), we obtain $\alpha^* = 0.029$. The corresponding point is $x^{(2)} = (3.018, 2.064)^T$ with an objective function value $f(x^{(2)}) = 0.107$.

This process continues until a point with a small derivative of the objective function is found. If the intermediate points fall on the constraint boundary frequently, this method may

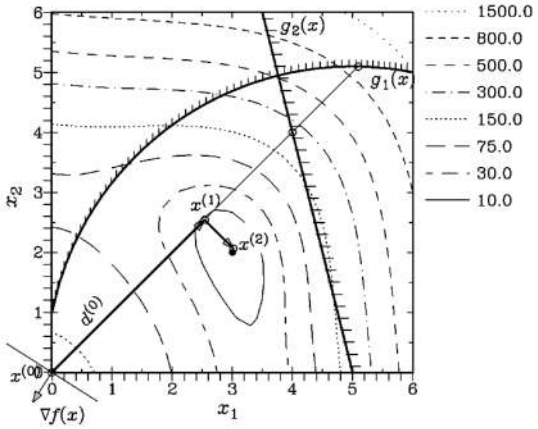

Figure 4.5: A number of iterations of the feasible direction method.

be expensive in terms of the overall computational time required to solve the problem. This situation may happen for problems with a narrow feasible region.

4.5. Quadratic Approximation Method

- Quadratic approximation to objective functions
- Linear approximations to constraints to make the calculations easier

Algorithm:

Step 1: Given $x^{(0)}$ and a suitable method for solving QP problem

Step 2: Formulate QP

$$\text{Minimize} \quad \nabla f(x^{(t)})^T d + \frac{1}{2} d^T \nabla^2 f(x^{(t)}) d$$

$$\text{Subject to} \quad h_k(x^{(t)}) + \nabla h_k(x^{(t)})^T d = 0 \qquad k = 1,2\ldots K$$
$$g_j(x^{(t)}) + \nabla g_j(x^{(t)})^T d \geq 0 \qquad j = 1,2\ldots J$$

Step 3: Solve the QP problem and set $x^{(t+1)} = x^{(t)} + d$

Step 4: Check for convergence. If not converged, Go to step 1

- Lead to nonvertex solutions
- The term $\nabla^2 f(x^{(t)})$ in the objective function may be replaced by a Lagrangian term or a variable-metric approximation for Hessian for easier computation

This procedure can be repeatedly used to solve general non-linear programming problems by formulating a QP problem at the current best solution. This methodology is called Sequential Quadratic Programming or SQP. Many commercial optimization software implements this algorithm in which the resulting QP problem is solved using a quasi-Newton multi-variable optimization algorithm.

4.6. References

[4.1] Box, M. J. (1965): A **new method of constrained** optimization **and a comparison with other methods.** Computer Journal. 8, 42-52.

[4.2] Kelly, J. E. (1960): **The cutting plane method for solving** convex **programs.** SIAM Journal. 8, 703-712.

[4.3] Luus, R. and Jaakola, T. H. I. (1973): **Optimization by direct** search **and systematic reduction of the size of search region.** AIChE Journal. 19,760-766.

[4.4] Mangasarian, 0. L. (1969): **Nonlinear Programming.** New York: McGrawHill.

[4.5] Rao, S. S. {1984): **Optimization Theory and** Applications**.** New Delhi:Wiley Eastern.

[4.6] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983): Engineering **Optimization-Methods and Applications.** New York: Wiley.

[4.7] Strang, G. (1980): **Linear Algebra and Its Applications.** Orlando: Academic Press.

[4.8] Taha, H. A. (1989): **Operations Research.** New York: MacMillan.

[4.9] Zangwill, W. I. (1969): **Nonlinear Programming.** Englewood Cliffs, New Jersey: Prentice-Hall.

[4.10] Zoutendijk, G. (1960): **Methods of Feasible Directions.** Amsterdam: Elsevier.

4.7. Questions

1. What are the Kuhn-Tucker conditions? How are they used?
2. Explain the concept of penalty function!
3. Explain the method of feasible directions!

5. NONTRADITIONAL OPTIMIZATION TECHNIQUES

This chapter makes a brief description of a nontraditional search and optimization methods. Further details will be supplied by separate sources.

5.1. Genetic Algorithms

Genetic algorithms (abbreviated as GAs) are computerized search and optimization algorithms designed based on the mechanics of natural genetics and natural selection.

5.1.1. Fundamental Differences with Traditional methods

- GAs work on a coding of parameters, instead of parameters. GAs exploit the coding similarities to achieve a parallel search.
- GAs work on a population of points, instead of a single point. That is why GAs are likely to find the global solutions.
- GAs do not require any derivative or auxiliary information. This extends the application of GAs to a wide variety of problem domains. That is why GAs are robust.
- GAs use probabilistic transition rules, instead of deterministic transition rules. This reduces the bias in the search. Initially the search direction is random and as iteration progresses, GAs obtain a directed search adaptively.

Algorithm:

Step 1: Choose a coding to represent problem parameters, a selection operator, a crossover operator, and a mutation operator. Choose population size, *N*, crossover probability, p_c, mutation probability, p_m. Initialize a random population of strings of size N. Set $t = 0$.

Step 2: Evaluate each string in the population.

Step 3: If $t > t_{\max}$ or other termination criteria is satisfied, **Terminate.**

Else Go to step 4.

Step 4: Reproduction on the population.

Step 5: Crossover on random pairs of strings.

Step 6: Mutation on every string.

Step 7: Set $t = t + 1$ and Go to step 2.

5.1.2. Reproduction Operator

Selects good strings from a population. Some of popular reproduction operators are as follows:

Proportionate Selection Strings are selected according to their fitness. Specifically, a string with fitness f_i is allocated $\frac{f_i}{f_{avg}}$ number of copies. Better strings are allocated more copies under this scheme.

Tournament Selection Usually, s strings are selected at random from a population, and the best is chosen. This procedure is continued until the whole population is filled up. This scheme can be performed with and without replacement. When performed without replacement, the best string gets s copies.

Ranking Selection All strings are first ranked from best to worst and ranked in a way so that the best gets s copies and the worst gets zero copies. Each string is then selected with a probability depending on its rank in the population.

5.1.3. Crossover Operator

Exchanges information between two strings selected at random. Some popular operators are as follows:

Single-point Crossover A cross site is chosen at random. The contents on one side of the site are exchanged between parent strings:

$$\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow \begin{array}{cc|ccc} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}$$

Two-point Crossover Two cross sites are chosen at random. The contents between two sites are exchanged between parent strings:

$$\begin{array}{c|cc|cc} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow \begin{array}{c|cc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{array}$$

Uniform Crossover Each bit-position is exchanged between parent strings with a probability 0.5:

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow \begin{array}{ccccc} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{array}$$

The search power of uniform crossover is most among these three operators, but the destruction probability of good partial substrings in parent strings is also more in uniform crossover.

In order to preserve some previously obtained good solutions, all strings in the population is not used in crossover. The proportion of population used in crossover is known as probability of crossover. Usually, a proportion of 0.60-0.95 is used.

5.1.4. Mutation Operator

Alters a bit value to another with a small probability. In a binary string, mutation operation is shown below:

$$11111 \Rightarrow 11101$$

Mutation maintains diversity in the population.

– Other advanced operators exist and tried

The objective is to minimize the function

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

in the interval $0 \le x_1, x_2 \le 6$. Recall that the true solution to this problem is $(3, 2)^T$ having a function value equal to zero.

Step 1: In order to solve this problem using genetic algorithms, we choose binary coding to represent variables x_1 and x_2. In the calculations here, 10-bits are chosen for each variable, thereby making the total string length equal to 20. With 10 bits, we can get a solution accuracy of $(6-0)/(2^{10}-1)$ or 0.006 in the interval (0,6). We choose roulette-wheel selection, a single-point crossover, and a bit-wise mutation operator. The crossover and mutation probabilities are assigned to be 0.8 and 0.05, respectively. We decide to have 20 points in the population. We set $t_{max} = 30$ and initialize the generation counter $t = 0$.

	String		x_2	x_1	f(x)	F (x)	Expected	Probability	Cumulative	Random	String	Actual	Mating	
	Substring-2	Substring-1					Count	of copying	probability	number	number	count	pool	
1	1110010000	1100100000	5.349	4.692	959.680	0.001	0.13	0.007	0.007	0.472	10	0	0010100100	1010101010
2	0001001101	0011100111	0.452	1.355	105.520	0.009	1.10	0.055	0.062	0.108	3	1	1010100001	0111001000
3	1010100001	0111001000	3.947	2.674	126.685	0.008	0.98	0.049	0.111	0.045	2	1	0001001101	0011100111
4	1001000110	1000010100	3.413	3.120	65.026	0.015	1.85	0.093	0.204	0.723	14	2	1110011011	0111000010
5	1100011000	1011100011	4.645	4.334	512.197	0.002	0.25	0.013	0.217	0.536	10	0	0010100100	1010101010
6	0011100101	0011111000	1.343	1.455	70.868	0.014	1.71	0.086	0.303	0.931	19	2	0011100010	1011000011
7	0101011011	0000000111	2.035	0.041	88.273	0.011	1.34	0.067	0.370	0.972	19	1	0011100010	1011000011
8	1110101000	1110101011	5.490	5.507	1436.563	0.001	0.12	0.006	0.376	0.817	17	0	0111000010	1011000110
9	1001111101	1011100111	3.736	4.358	265.556	0.004	0.49	0.025	0.401	0.363	7	1	0101011011	0000000111
10	0010100100	1010101010	0.962	4.000	39.849	0.024	2.96	0.148	0.549	0.189	4	3	1001000110	1000010100
11	1111101001	0001110100	5.871	0.680	814.117	0.001	0.14	0.007	0.556	0.220	6	0	0011100101	0011111000
12	0000111101	0110011101	0.358	2.422	42.598	0.023	2.84	0.142	0.698	0.288	6	3	0011100101	0011111000
13	0000111110	1110001101	0.364	5.331	318.746	0.003	0.36	0.018	0.716	0.615	12	1	0000111101	0110011101
14	1110011011	0111000010	5.413	2.639	624.164	0.002	0.24	0.012	0.728	0.712	13	1	0000111110	1110001101
15	1010111010	1010111000	4.094	4.082	286.800	0.003	0.37	0.019	0.747	0.607	12	0	0000111101	0110011101
16	0100011111	1100111000	1.683	4.833	197.556	0.005	0.61	0.030	0.777	0.192	4	0	1001000110	1000010100
17	0111000010	1011000110	2.639	4.164	97.699	0.010	1.22	0.060	0.837	0.386	9	1	1001111101	1011100111
18	1010010100	0100001001	3.871	1.554	113.201	0.009	1.09	0.054	0.891	0.872	18	1	1010010100	0100001001
19	0011100010	1011000011	1.326	4.147	57.753	0.017	2.08	0.103	0.994	0.589	12	2	0000111101	0110011101
20	1011100011	1111010000	4.334	5.724	987.955	0.001	0.13	0.006	1.000	0.413	10	0	0010100100	1010101010

Table 5.1: Evaluation and reproduction of a random population is illustrated.

Step 2: The next step is to evaluate each string in the population. We calculate the fitness of the first string. The first substring (1100100000) decodes to a value equal to $\left(2^9+2^8+2^5\right)$ or 800. Thus, the corresponding parameter value is equal to $0+(6-0)\times 800/1023$ or 4.692. The second substring (1110010000) decodes to a value equal to $\left(2^9+2^8+2^7+2^4\right)$ or 912. Thus, the corresponding parameter value is equal to $0+(6-0)\times 912/1023$ or 5.349. Thus, the first string corresponds to the point $x^{(1)}=(4.692, 5.349)^T$. These values can now be substituted in the objective function expression to obtain the function value. It is found that the function value at this point is equal to $f(x^{(1)})=959.680$. We now calculate the fitness function value at this point using the transformation rule: $F(x^{(1)})=1.0/(1.0+959.680)=0.001$. This value is used in the reproduction operation. Similarly, other strings in the population are evaluated and fitness values are calculated. Table 5.1 shows the objective function value and the fitness value for all 20 strings in the initial population.

Step 3: Since $t=0<t_{\max}=30$, we proceed to Step 4.

Step 4: At this step, we select good strings in the population to form the mating pool. In order to use the roulette-wheel selection procedure, we first calculate the average fitness of the population. By adding the fitness values of all strings and dividing the sum by the population size, we obtain $\overline{F}=0.008$. The next step is to compute the expected count of each string as $F(x)/\overline{F}$. The values are calculated and shown in column A of Table 5.1. In other words, we can compute the probability of each string being copied in the mating pool by dividing these numbers with the population size (column B). Once these probabilities are calculated, the cumulative probability can also be computed. These distributions are also shown in column C of Table 5.1. In order to form the mating pool, we create random numbers between zero and one (given in column D) and identify the particular string which is specified by each of these random numbers. For example, if the random number 0.472 is created, the tenth string gets a copy in the mating pool, because that string occupies the interval (0.401, 0.549), as shown in column C. Column E refers to the selected string. Similarly, other strings are selected according to the random numbers shown in column D. After this selection procedure is repeated n times (n is the population size), the number of selected copies for each string is counted. This number is shown in column F. The complete mating pool is also shown in the table. Columns A and F reveal that the theoretical expected count and the true count of each string more or less agree with each other. Figure 5.1 shows the initial random population and the mating pool after reproduction. The points marked with an enclosed box are the points in the mating pool. The action of the reproduction operator is clear from this plot. The inferior points have been probabilistically eliminated from further consideration. Notice that not all selected points are better than all rejected points. For example, the 14th individual (with a fitness value 0.002) is selected but the 16th individual (with a function value 0.005) is not selected.

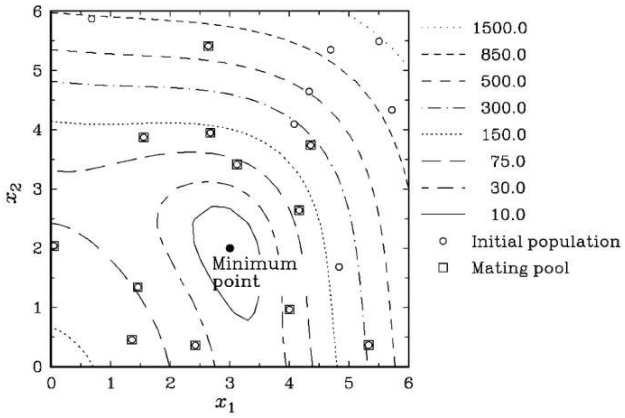

Figure 5.1.: The initial population (marked with empty circles) and the mating pool (marked with boxes) on a contour plot of the objective function. The best point in the population has a function value 39.849 and the average function value of the initial population is 360.540.

Although the above roulette-wheel selection is easier to implement, it is noisy. A more stable version of this selection operator is sometimes used. After the expected count for each individual string is calculated, the strings are first assigned copies exactly equal to the mantissa of the expected count. Thereafter, the regular roulette-wheel selection is implemented using the decimal part of the expected count as the probability of selection.

This selection method is less noisy and is known as the stochastic remainder selection.

Step 5: At this step, the strings in the mating pool are used in the crossover operation. In a single-point crossover, two strings are selected at random and crossed at a random site. Since the mating pool contains strings at random, we pick pairs of strings from the top of the list. Thus, strings 3 and 10 participate in the first crossover operation. When two strings are chosen for crossover, first a coin is flipped with a probability $p_c = 0.8$ to check whether a crossover is desired or not. If the outcome of the coin-flipping is true, the crossing over is performed, otherwise the strings are directly placed in an intermediate population for subsequent genetic operation. It turns out that the outcome of the first coin-flipping is true, meaning that a crossover is required to be performed. The next step is to find a cross-site at random. We choose a site by creating a random number between $(0, \ell - 1)$ or (0, 19). It turns out that the obtained random number is 11. Thus, we cross the strings at the site 11 and create two new strings. After crossover, the children strings are placed in the intermediate population. Then, strings 14 and 2 (selected at random) are used in the crossover operation. This time the coin-flipping comes true again and we perform the crossover at the site 8 found at random. The new children strings are put into the intermediate population. Figure 5.2 shows how points cross over and form new points. The points marked with a small box are the points in the mating pool and the points marked with a small circle are children points created after crossover operation. Notice that not all 10 pairs of points in the mating pool cross with each other. With the flipping of a coin with a probability $p_c = 0.8$, it turns out that fourth, seventh, and tenth crossovers come out to be false. Thus, in these cases, the strings are copied directly into the intermediate population. The complete population at the end of the crossover operation is shown in Table 5.2.

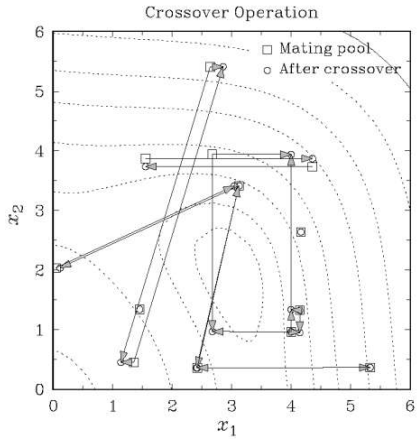

Figure 5.2.: The population after the crossover operation. Two points are crossed over to form two new points. Of ten pairs of strings, seven pairs are crossed.

It is interesting to note that with $p_c = 0.8$, the expected number of crossover in a population of size 20 is $0.8 \times 20/2$ or 8. In this exercise problem, we performed seven crossovers and in three cases we simply copied the strings to the intermediate population. Figure 5.2 shows that some good points and some not-so-good points are created after crossover. In some cases, points far away from the parent points are created and in some cases points close to the parent points are created.

0010100100 1010101010	yes	9	0010100101 0111001000	0010101101 0111001000	1.015	2.674	18.886	0.050
1010100001 0111001000	yes	9	1010100000 1010101010	1010100001 1010101010	3.947	4.000	238.322	0.004
0001001101 0011100111	yes	12	0001001101 0011000010	0001001101 0001000010	0.452	0.387	149.204	0.007
1110011011 0111000010	yes	12	1110011011 0111100111	1110011011 0101100011	5.413	2.082	596.340	0.002
0010100100 1010101010	yes	5	0010100010 1011000011	0010100010 1011000011	0.950	4.147	54.851	0.018
0011100010 1011000011	yes	5	0011100100 1010101010	0011100100 1110101010	1.337	5.501	424.583	0.002
0011100010 1011000011	no		0011100010 1011000011	0011100011 1011100011	1.331	4.334	83.929	0.012
0111000010 1011000110	no		0111000010 1011000110	0101010010 1011000110	1.982	4.164	70.472	0.014
0101011011 0000000111	yes	14	0101011011 0000010100	0101011011 0000010100	2.035	0.117	87.633	0.011
1001000110 1000010100	yes	14	1001000110 1000000111	1001010110 1000000111	3.507	3.044	72.789	0.014
0011100101 0011111000	yes	1	0011100101 0011111000	0011100101 0011111000	1.343	1.455	70.868	0.014
0011100101 0011111000	yes	1	0011100101 0011111000	0011100101 0011111000	1.343	1.455	70.868	0.014
0000111101 0110011101	no		0000111101 0110011101	0000101101 0111011100	0.264	2.792	25.783	0.037
0000111110 1110001101	no		0000111110 1110001101	0000111110 1110001101	0.364	5.331	318.746	0.003
0000111101 0110011101	yes	18	0000111101 0110011100	0000111101 0110011100	0.358	2.416	42.922	0.023
1001000110 1000010100	yes	18	1001000110 1000010101	1001000110 0000010101	3.413	0.123	80.127	0.012
1001111101 1011100111	yes	10	1001111101 0100001001	1001111101 0100001001	3.736	1.554	95.968	0.010
1010010100 0100001001	yes	10	1010010100 1011100111	1010010100 1010100111	3.871	3.982	219.426	0.005
0000111101 0110011101	no		0000111101 0110011101	0000111101 0110011101	0.358	2.422	42.598	0.023
0010100100 1010101010	no		0010100100 1010101010	0010100100 1010101010	0.962	4.000	39.849	0.024

Table 5.2: Crossover and mutation operators are shown.

Step 6: The next step is to perform mutation on strings in the intermediate population.

For bit-wise mutation, we flip a coin with a probability $p_m = 0.05$ for every bit. If the outcome is true, we alter the bit to 1 or 0 depending on the bit value. With a probability of 0.05, a population size 20, and a string length 20, we can expect to alter a total of about $0.05 \times 20 \times 20$ or 20 bits in the population. Table 5.2 shows the mutated bits in bold characters in the table. As counted from the table, we have actually altered 16 bits. Figure 6.3 shows the effect of mutation on the intermediate population. In some cases, the mutation operator changes a point locally and in some other it can bring a large change. The points marked with a small circle are points in the intermediate population. The points marked with a small box constitute the new population (obtained after reproduction, crossover, and mutation). It is interesting to note that if only one bit is mutated in a string, the point is moved along a particular variable only. Like the crossover operator, the mutation operator has created some points better and some points worse than the original points. This flexibility enables GA operators to explore the search space properly before converging to a region prematurely. Although this requires some extra computation, this flexibility is essential to solve global optimization problems.

Step 7: The resulting population becomes the new population. We now evaluate each string as before by first identifying the substrings for each variable and mapping the decoded values of the substrings in the chosen intervals. This completes one iteration of genetic algorithms. We increment the generation counter to $t = 1$ and proceed to Step 3 for the next iteration. The new population after one iteration of GAs is shown in Figure 5.3 (marked with empty boxes). The figure shows that in one iteration, some good points have been found. Table 5.2 also shows the fitness values and objective function values of the new population members.

The average fitness of the new population is calculated to be 0.015, a remarkable improvement from that in the initial population (recall that the average in the initial population was 0.008).

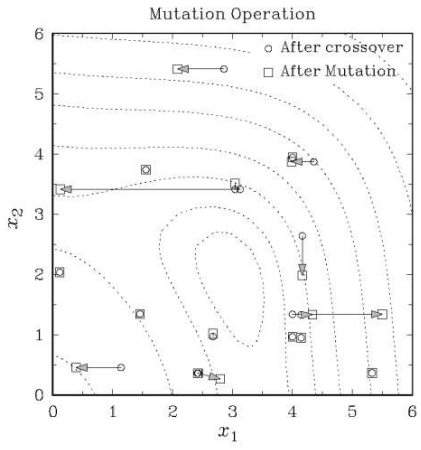

Figure 5.3.: The population after mutation operation. Some points do not get mutated and remain unaltered. The best point in the population has a function value 18.886 and the average function value of the population is 140.210, an improvement of over 60 per cent.

The best point in this population is found to have a fitness equal to 0.050, which is also better than that in the initial population (0.024). This process continues until the maximum allowable generation is reached or some other termination criterion is met. The population after 25 generation is shown in Figure 5.4. At this generation, the best point is found to be $(3.003, 1.994)^T$ with a function value 0.001. The fitness value at this point is equal to 0.999 and the average population fitness of the population is 0.474. The figure shows how points are clustered around the true minimum of the function in this generation. A few inferior points are still found in the plot. They are the result of some unsuccessful crossover events. We also observe that the total number of function evaluations required to obtain this solution is $0.8 \times 20 \times 26$ or 416 (including the evaluations of the initial population).

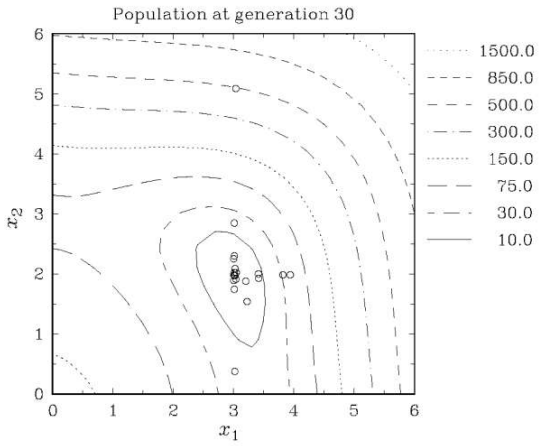

Figure 5.4.: All 20 points in the population at generation 25 shown on the contour plot of the objectivefunction. The figure shows that most points are clustered around the true minimum.

In order to show the efficiency of GAs in arriving at a point close to the true optimum, we perform two more simulations starting with different initial populations. Figure 6.5 shows how the function value of the best point in a population reduces with generation number. Although all three runs have a different initial best point, they quickly converge to a solution close to the true optimum (recall that the optimum point has a function value equal to zero).

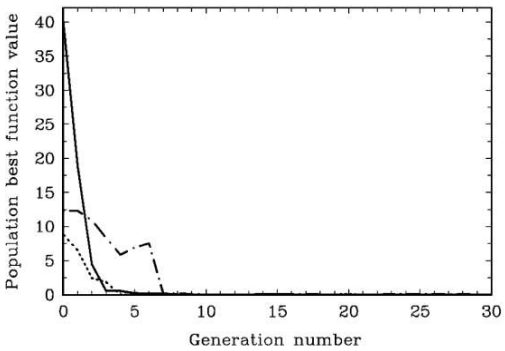

Figure 5.5: The function value of the best point in the population for three independent GA runs. All runs quickly converge to a point near the true optimum.

In order to illustrate the schema processing through genetic operators, we investigate the growth of a particular schema $H = (0 * ... * \ * ... *)$. This schema represents all points in the range $0 \le x_2 < 3$. The optimum point lies in this region. With reference to Equation (6.3), we observe that the order, defining length, and the fitness of the schema are such that it is a building block. This schema contains more good points than it's competitor $H^c = (1 * ... * \ * ... *)$ which represents the range $3 \le x_2 < 6$. According to Equation (6.3), the schema H must increase exponentially due to the action of genetic operators. We observe that in the random initial population the schema H has nine strings and the schema H^c has 11 strings. At the end of one generation, the population has 14 strings representing the schema H and only six strings representing the schema H^c. We may also investigate other interesting regions in the search space and observe their growth in terms of the number of representative points in the population. Other low-order and above-average schemata are also processed similarly and are combined to form higher-order and good schemata. This processing of several schemata happens in parallel without any extra book-keeping (Goldberg, 1989). Eventually, this processing forms the optimum or a near-optimum point.

5.2. References

[5.1] Goldberg, D. E. (1989): **Genetic Algorithms in Search, Optimization, and Machine Learning.** Reading, Mass.: Addison-Wesley.

5.3. Questions

1. How does a Genetic algorithm work? What are the main operators?
2. What is the aim of the crossover operator?
3. What are the advantages/disadvantages of GAs?

6. MULTI-CRITERION OPTIMIZATION

As the name suggests, a multi-objective optimization problem (MOOP) deals with more than one objective function. In most practical decision-making problems, multiple objectives or multiple criteria are evident. Because of a lack of suitable solution methodologies, an MOOP has been mostly cast and solved as a single objective optimization problem in the past. However, there exist a number of fundamental differences between the working principles of single and multi-objective optimization algorithms. In a single-objective optimization problem, the task is to find one solution (except in some specific multi-modal optimization problems, where multiple optimal solutions are sought) which optimizes the sole objective function.
Extending the idea to multi-objective optimization, it may be wrongly assumed that the task in a multi-objective optimization is to find an optimal solution corresponding to each objective function. In this chapter, we will discuss the principles of multi-objective optimization and present optimality conditions for any solution to be optimal in the presence of multiple objectives.

6.1. Multi-Objective Optimization Problem

A multi-objective optimization problem has a number of objective functions which are to be minimized or maximized. As in the single-objective optimization problem, here too the problem usually has a number of constraints which any feasible solution (including the optimal solution) must satisfy. In the following, we state the multi-objective optimization problem (MOOP) in its general form:

$$min/max \quad f_m(x), \qquad m = 1,2 \ldots, M;$$

subject to:

$$g_j(x) \geq 0, j = 1,2, \ldots, J; \qquad (6.1)$$

$$h_k(x) = 0 \qquad k = 1,2, \ldots, K;$$

$$\chi_i^{(L)} \leq \chi_i \leq \chi_i^{(U)} \qquad i = 1,2, \ldots, n;$$

A solution $\mathbf{x}$ is a vector of n decision variables: $\mathbf{x} = (\chi_1, \chi_2, \ldots, \chi_n)^T$. The last set of constraints are called variable bounds, restricting each decision variable χ_i to take value within a lower $\chi_i^{(L)}$ and an upper $\chi_i^{(U)}$ bound. These bounds constitute a decision variable space D, or simply the decision space. Throughout this chapter, we use the terms point and solution interchangeably to mean a solution vector $\mathbf{x}$. Associated with the problem are J inequality and K equality constraints. The terms $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ are called *constraint functio*ns. The inequality constraints are treated as 'greater-than-equal-to' types, although a 'less-than-equal-to' type inequality constraint is also taken care of in the above formulation. In the latter case, the constraint must be converted into a 'greater-than-equal-to' type constraint by multiplying the constraint function by –l (Deb 1995). A solution $\boldsymbol{x}$ that does not satisfy all of the $(J + K)$ constraints and all of the $2N$ variable bounds stated above is called an infeasible solution. On the other hand, if any solution $\mathbf{x}$ satisfies all constraints and variable bounds, it is known as a feasible solution. Therefore, we realize that in the presence of constraints, the entire decision variable space D need not be feasible. The set of all feasible solutions is called the feasible

region, or *S*. In this script, sometimes we will refer to the feasible region as simply the search space.
There are M objective functions $f(x) = (f_1(x), f_2(x), \ldots f_M(x))^T$ considered in the above formulation. Each objective function can be either minimized or maximized. The duality principle (Deb,1995, Rao, Reklaitis et al.) in the context of optimization, suggests that we can convert a maximization problem into a minimization one by multiplying the objective function by -1. The duality principle has made the task of handling mixed type of objectives much easier. Many optimization algorithms are developed to solve only one type of optimization problems, such as e.g. minimization problems. When an objective is required to be maximized by using such an algorithm, the duality principle can be used to transform the original objective for maximization into an objective for minimization.
Although there is a difference in the way that a criterion function and an objective function is defined (Chankong), in a broad sense we treat them here as identical. One of the striking differences between single-objective and multi-objective optimization is that in multi-objective optimization the objective functions constitute a multi-dimensional space, in addition to the usual decision variable space.
This additional space is called the objective space *Z*. For each solution **x** in the decision variable space, there exists a point in the objective space, denoted by $f(x) = z = (z_1, z_2, \ldots, z_M)^T$. The mapping takes place between an *n* dimensional solution vector and an M-dimensional objective vector. Figure 6.1 illustrates these two spaces and a mapping between them.

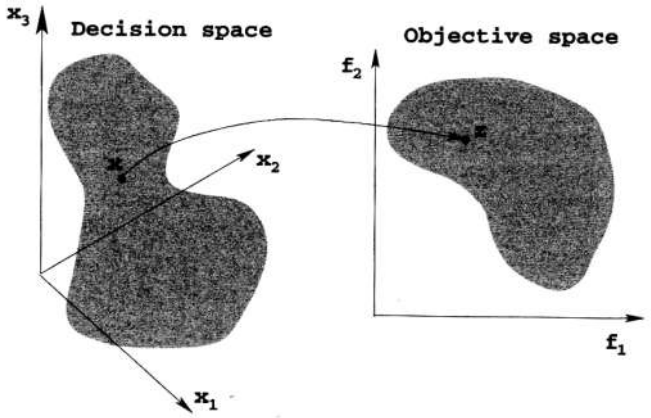

Figure 6.1.: Representation of the decision variable space and the corresponding objective space

Multi-objective optimization is sometimes referred to as vector optimization, because a vector of objectives, instead of a single objective, is optimized.

6.2. Principles of Multi-Objective Optimization

We illustrate the principles of multi-objective optimization through an airline routing problem. We all are familiar with the intermediate stopovers that most airlines force us to take, particularly when flying long distance. Airlines try different strategies to compromise on the number of intermediate stopovers and earn a large business mileage by introducing 'direct' flights. Let us take a look at a typical, albeit hypothetical, airline routing for some cities in the United States of America, as shown in Figure 6.2.

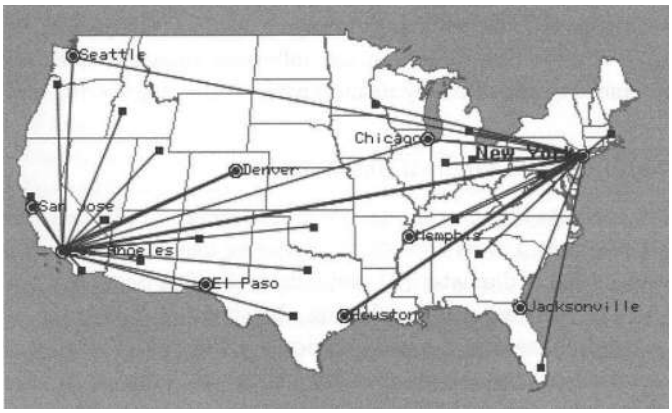

Figure 6.2.: A typical network of airline routes showing hub-like connections

If we look carefully, it is evident that there are two main 'hubs' (Los Angeles and New York) for this airline. If these two hubs are one's cities of origin and destination, the traveler is lucky. This is because there are likely to be densely packed schedules of flights between these two cities. However, if one has to travel between some other cities, let us say between Denver and Houston, there is no direct flight. The passenger has to travel to one of these hubs and then take more flights from there to reach the destination. In the Denver-Houston case, one has to fly to Los Angeles, fly on to New York and then make the final lap to Houston.

To an airline, such modular networks of routes is easiest to maintain and coordinate. Better service facilities and ground staff need only be maintained at the hubs, instead of at all airports. Although one then travels longer distance than the actual geographical distance between the cities of origin and destination, this helps an airline to reduce the cost of its operation. Such a solution is ideal from the airline's point of view, but not so convenient from the point of view of a passenger's comfort. However, the situation is not that biased against the passenger's point of view either. By reducing the cost of operation, the airline is probably providing a cheaper ticket. However, if comfort or convenience is the only consideration to a passenger, the latter would like to have a network of routes which would be entirely different to that shown in Figure 6.1.

A hypothetical routing is shown in Figure 6.2. In such a network, any two airports would be connected by a route, thereby allowing a direct flight between all airports. Since the operation cost for such a scenario will be exorbitantly high, the cost of flying with such a network would also be high.

Thus, we see a trade-off between two objectives in the above problem-cost versus convenience. A less-costly flight is likely to have more intermediate stopovers causing more inconvenience to a passenger, while a high-comfort flight is likely to have direct routes, thus causing an expensive ticket. The important matter is that between any two arbitrary cities in the first map (which resembles the routing of most airlines) there does

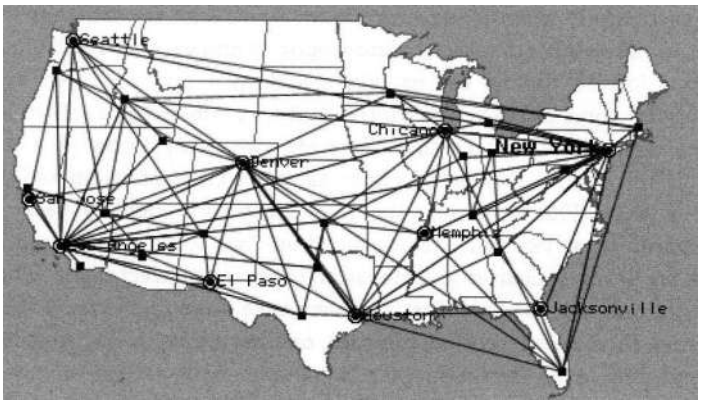

Figure 6.3.: A hypothetical (but convenient) airline routing

not exist a flight which is less costly as well as being largely convenient. If there was, that would have been the only solution to this problem and nobody would have complained about paying more or wasting time by using a 'hopping' flight. The above two solutions of a hub-like network of routes and a totally connected network of routes are two extreme solutions to this two-objective optimization problem. There exist many other compromised solutions which have lesser hub-like network of routes and more expensive flights than the solution shown in Figure 6.1.

Innovative airlines are constantly on the lookout for such compromises and in the process making the network of routes a bit less hub-like, so giving the passengers a bit more convenience. The 'bottom-line' of the above discussion is that when multiple conflicting objectives are important, there cannot be a single optimum solution which simultaneously optimizes all objectives. The resulting outcome is a set of optimal solutions with a varying degree of objective values. In the following subsection, we will make this qualitative idea more quantitative by discussing a simple engineering design problem.

6.3. Illustrating Pareto-Optimal Solutions

We take a more concrete engineering design problem here to illustrate the concept of Pareto-optimal solutions. Let us consider a cantilever design problem (Figure 6.4) with two decision variables, i.e. diameter (d) and l length

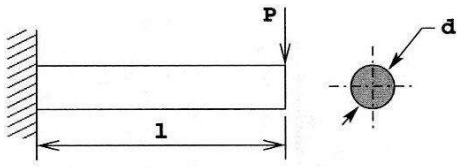

Figure 6.4.: A schematic of a cantilever beam.

The beam has to carry an end load P. Let us also consider two conflicting objectives of design, i.e. minimization of weight f_1 and minimization of end deflection f_2. The first objective will resort to an optimum solution having the smaller dimensions of d and l, so that the overall weight of the beam is minimum. Since the dimensions are small, the beam will not be adequately rigid and the end deflection of the beam will be large. On the other hand, if the beam

is minimized for end deflection, the dimensions of the beam are expected to be large, thereby making the weight of the beam large. For our discussion, we consider two constraints: the developed maximum stress σ_{max} is less than the allowable strength σ_y and the end deflection δ is smaller than a specified limit σ_{max}. With all of the above considerations, the following two-objective optimization problem is formulated as follows:

$$\min f_1(d,l) = \rho \frac{\pi d^2}{4} l$$
$$\min f_2(d,l) = \delta = \frac{64Pl^3}{3E\pi d^4}$$

subject to

$$\sigma_{max} \le S_y$$
$$d \le d_{max},$$

where the maximum stress is calculated as follows:

$$\sigma_{max} = \frac{32Pl}{\pi d^3}$$

The following parameter values are used:

ρ = 7800 kg/m^3, P = 1kN, E = 210 GPa,
S_y = 300 MPa, δ_{max} = 5mm.

The left plot in Figure 6.5. marks the feasible decision variable space in the overall search space enclosed by $10 \le d \le 50$mm and $200 \le l \le 1000$mm. It is clear that all solutions in the rectangular decision space are feasible. Every feasible solution in this space can be mapped to a solution in the feasible objective space shown in the right plot. The correspondence of a point in the left figure with that in the right figure is also shown.

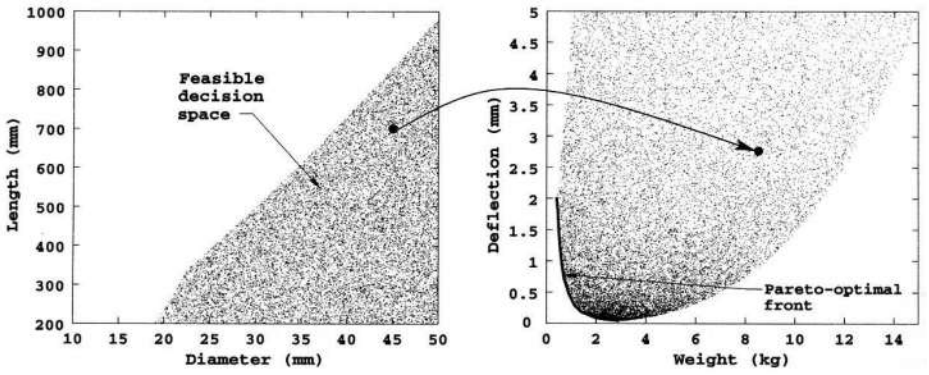

Figure 6.5.: The feasible decision variable space (left) and the feasible objective space (right).

This figure shows many solutions trading-off differently between the two objectives. Any two solutions can be picked from the feasible objective space and compared. For some pairs of solutions, it can be observed that one solution is better than the other objectives. For certain

other pairs, it can be observed that one solution is better than the other in one objective, but is worse in the second objective. In order to establish which solution(s) are optimal with respect to both objectives, let us hand-pick few solutions from the search space. Figure 6.6 is drawn with many such solutions, and five of these solutions (marked A to *E*) are presented in Table 1. Of these solutions, the minimum weight solution (*A*) has a diameter of 18.94 mm, while the minimum deflection solution (*D*) has a diameter of 50 mm. It is clear that solution *A* has a smaller weight, but has a larger end-deflection than solution *D*. Hence, none of these two solutions can be said to be better than the other with respect to both objectives. When this happens between two solutions, they are called *non-dominated solutions*. If both objectives are equally important, one cannot say, for sure, which of these two solutions is better with respect to both objectives. Two other similar solutions (*B* and *C*) are also shown in the figure and in the tables. Of these four solutions (*A* to *D*), any pair of solutions can be compared with respect to both objectives. Superiority of one over the other cannot be established with both objectives in mind. There exist many such solutions in the search space.
For clarity, these solutions are joined with a curve in the figure. All solutions lying on this curve are special in the context of multi-objective optimization and are called *Pareto-optimal solutions*. The curve formed by joining these solutions is known as a *Pareto-optimal front*. The same Pareto-optimal front is also marked on the right plot of Figure 6.5 by a continuous curve. It is interesting to observe that this front lies in the bottom-left corner of the search space for problems where all objectives are to be minimized.

Solution	d (mm)	l (mm)	Weight (kg)	Deflection (mm)
A	18.94	200.00	0.44	2.04
B	21.24	200.00	0.58	1.18
C	34.19	200.00	1.43	0.19
D	50.00	200.00	3.06	0.04
E	33.02	362.49	2.42	1.31

Table 6.1 Five solutions for the cantilever design problem.

It is important to note that the feasible objective space not only contains Pareto-optimal solutions, but also solutions that are not optimal. The entire feasible search space can be divided into two sets of solutions - a Pareto-optimal and a non-Pareto-optimal set. Consider solution *E* in Figure 6.6 and also in Table 1.

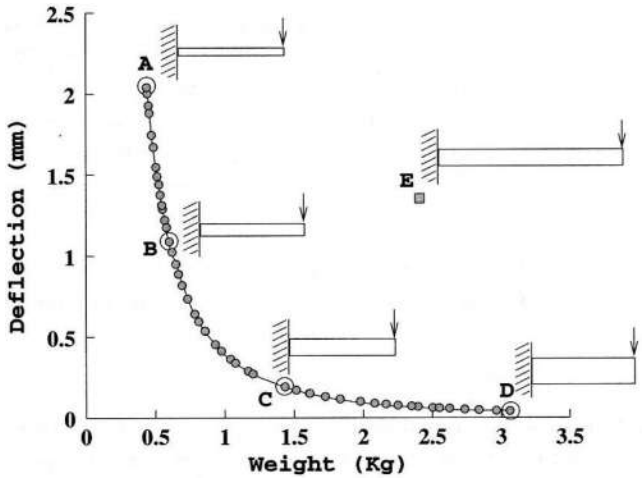

Figure 6.6.: Four Pareto-optimal solutions and one non-optimal solution.

By comparing this with solution *C*, we observe that the latter is better than solution *E* in both objectives. Since solution *E* has a larger weight and a larger end-deflection than solution *C*, the latter solution is clearly the better of the two. Thus, solution E is a sub-optimal solution and is of no interest to the user. When this happens in comparison of two solutions, solution *C* is said to *dominate* solution E or that solution E is *dominated* by solution *C*. There exist many such solutions in the search space which can be dominated by at least one solution from the Pareto-optimal set.

In other words, there exists at least one solution in the Pareto-optimal set, which will be better than any non-Pareto-optimal solution. It is clear from the above discussion that in multi-objective optimization the task is to find the Pareto-optimal solutions.

Instead of considering the entire search space for finding the Pareto- and non-Pareto- optimal sets, such a division based on domination can also be made for a finite set of solutions *P* chosen from the search space. Using a pair-wise comparison as above, one can divide the set *P* into two non-overlapping sets P_1 and P_2, such that P_1 contains all solutions that do not dominate each other and at least one solution in P_1 dominates any solution in P_2. The set P_1 is called the non-dominated set, while the set P_2 is called the *dominated* set. There is an interesting observation about dominated and non-dominated sets, which is worth mentioning here. Let us compare solutions *D* and *E*. Solution *D* is better in the second objective but is worse in the first objective compared to solution *E*. Thus, in the absence of solutions *A*, *B*, *C*, and any other non-dominated solution, we would be tempted to put solution *E* in the same group with solution *D*. However, the presence of solution *C* establishes the fact that solutions *C* and *D* are non-dominated with respect to each other, while solution *E* is a dominated solution. Thus, the non-dominated set must be collectively compared with any solution $\boldsymbol{x}$ for establishing whether the latter solution belongs to the non-dominated set or not. Specifically, the following two conditions must be true for a non-dominated set P_1:

1. Any two solutions of P_1 must be non-dominated with respect to each other.
2. Any solution not belonging to P_1 is dominated by at least one member of P_1.

6.4. Objectives in Multi-Objective Optimization

It is clear from the above discussion that, in principle, the search space in the context of multiple objectives can be divided into two non-overlapping regions, namely one which is optimal and one which is non-optimal. Although a two-objective problem is illustrated above, this is also true in problems with more than two objectives. In the case of conflicting objectives, usually the set of optimal solutions contains more than one solution. Figure 6.6 shows a number of such Pareto-optimal solutions denoted by circles.

In the presence of multiple Pareto-optimal solutions, it is difficult to prefer one solution over the other without any further information about the problem. If higher-level information is satisfactorily available, this can be used to make a biased search. However, in the absence of any such information, all Pareto-optimal solutions are equally important. Hence, in the light of the ideal approach, it is important to find as many Pareto-optimal solutions as possible in a problem. Thus, it can be conjectured that there are two goals in a multi-objective optimization:

1. To find a set of solutions as close as possible to the Pareto-optimal front.
2. To find a set of solutions as diverse as possible.

The first goal is mandatory in any optimization task. Converging to a set of solutions which are not close to the true optimal set of solutions is not desirable. It is only when solutions converge close to the true optimal solutions that one can be assured of their near-optimality properties. This goal of multi-objective optimization is common to the similar optimality goal in a single-objective optimization.
On the other hand, the second goal is entirely specific to multi-objective optimization. In addition to being converged close to the Pareto-optimal front, they must also be sparsely spaced in the Pareto-optimal region. Only with a diverse set of solutions, can we be assured of having a good set of trade-off solutions among objectives. Since MOEAs deal with two spaces - decision variable space and objective space 'diversity' among solutions can be defined in both of these spaces. For example, two solutions can be said to be diverse in the decision variable space if their Euclidean distance in the decision variable space is large. Similarly, two solutions are diverse in the objective space, if their Euclidean distance in the objective space is large. Although in most problems diversity in one space usually means diversity in the other space, this may not be so in all problems. In such complex and nonlinear problems, it is then the task to find a set of solutions having a good diversity in the desired space.

6.5. Non-Conflicting Objectives

It is worth pointing out that there exist multiple Pareto- optimal solutions in a problem only if the objectives are conflicting to each other. If the objectives are not conflicting to each other, the cardinality of the Pareto-optimal set is one. This means that the minimum solution corresponding to any objective function is the same. For example, in the context of the cantilever design problem, if one is interested in minimizing the end-deflection δ and minimizing the maximum developed stress in the beam, σ_{max}, the feasible objective space is different.

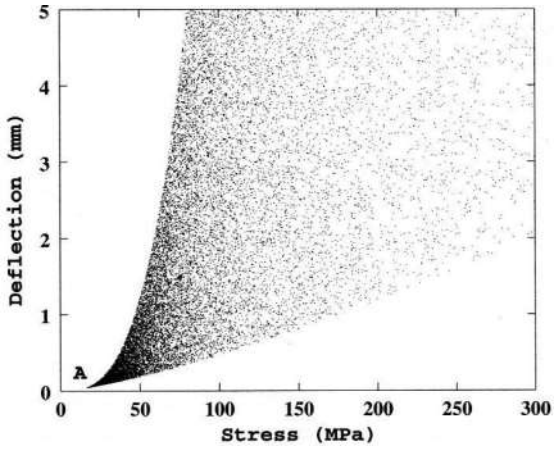

Figure 6.7.: End deflection and developed maximum stress are two non-conflicting objectives leading to one optimal solution (A).

Figure 6.7 shows that the Pareto-optimal front reduces to a single solution (solution *A* marked on the figure). A little thought will reveal that the minimum end-deflection happens for the most rigid beam with the largest possible diameter. Since this beam also corresponds to the smallest developed stress, this solution also corresponds to the minimum-stress solution. In certain problems, it may not be obvious that the objectives are not conflicting to each other. In such combinations of objectives, the resulting Pareto-optimal set will contain only one optimal solution.

6.6. References

[6.1] Deb, K. Optimization for Engineering Design: Algorithms and Examples. New Delhi: Pretience Hall.

[6.2] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983): Engineering Optimization-Methods and Applications. New York: Wiley.

[6.3] Rao, S. S. (1984): Optimization Theory and Applications. New Delhi:Wiley Eastern.

[6.4] Chankong, V., Haimes, Y. Y., Thadathil, J. and Zionts, S.: Multiple Criteria optimization: A state-of-the-art review. In Proceedings of the Sixth International Conference on Multiple-Criteria Decision Making, pp. 36-90

6.7. Questions

1. Explain the principles of multi-objective optimization!
2. What are the Pareto-optimal solutions?

7. OPTIMIZATION OF PRODUCTS AND MACHINE COMPONENTS

First we have introduced the basic element and formulations of an optimization problem and then discussed the solution techniques. In this chapter we are coming to the definition of the main problem classes and introducing the role and place of the different optimization techniques in the design process. Through industrial examples illustrates the power of the used methods solving mainly mechanical engineering problems.

The complex industrial optimization problems generally include behavior constraints, which can be evaluated only with numerical structural analysis techniques (Finite Element Method or Boundary Element Method). The studied problems in this chapter can be calculated using numerical techniques both for structural analysis and for optimization.

7.1. The place and role of the optimization in the design process

The design process may be divided into four stages:

1. **Formulation of functional requirements**, which is the first step in any design procedure. In some cases the functional requirements are not explicitly stated beforehand, and the designer has to investigate and take part in formulating these requirements. However, functional requirements are often established already before the engineer enters the design process.

2. The **conceptual design stage**, characterized by creativity, and engineering judgment of the designer, is a critical part of the design process. It deals with the overall planning of a system to serve its functional purposes. At this stage, the designer experiences the greatest challenges as well as chances of success or failure. Selection of the overall topology, type of structure, and materials are some of the decisions made by the designer at the conceptual design stage. In general, this part of the design process cannot be performed by a computer.

3. **Optimization**. Within a selected concept there may be many possible designs that satisfy the functional requirements, and a "trial-and-error" procedure may be employed to choose the optimal design. The computer is most suitable to carry out this part of using optimization methods to search for the optimal solutions. Thus, optimization in the present context is an automated design procedure giving the optimal values of certain design quantities, considering desired criteria and constraints.

4. **Detailing**. After completing the optimization stage, the results obtained must be checked and modified if necessary. In the final detailing stage, engineering judgment and experience are required, and it is again usually necessary for the designer to take part in the decision-making process**.**

Iterative procedures for the four stages are often required before the final solution is achieved, because the planning process in not a linear sequence. The portion of the structural design process that can be optimized automatically has been considerably increased in recent years. Optimization procedures are usually used to solve specific subproblems and the field of automated design is strongly connected with computer-aided design.

The optimization stage can also be divided into steps:

1. Formulating the optimization problem, choosing the design variables and design parameters.
2. Taking the assumptions.
3. Defining the goal function and the optimization constraints
4. Choosing the suitable optimization algorithm, and defining the convergence conditions.
5. Performing the calculation.
6. Comparing the results with the analytical ones (if exist), taking into the consideration the effect of the assumption.

The available methods of optimization may be subdivided into two categories:

Analytical methods are most suited for such fundamental studies of single structural components, but they are not able to handle larger structural systems. In analytic optimization problems the structural design is represented by a number of unknown functions and the goal is to find the form of these functions.

Numerical methods, which are usually employing a branch in the field of numerical mathematics called programming methods. The recent developments in this branch are closely related to the rapid growth in computing capacities affected by the development of computers. In the numerical methods, a near optimal design is automatically generated in an iterative manner. An initial guess is used as starting point for a systematic search for better designs. The search is terminated when certain criteria are satisfied; indicating that the current design is sufficiently close to the true optimum. Problems solved by numerical methods are called finite optimization problems. This is due to the fact that they can be formulated by a finite number of variables. The modern CAE systems inherit not only the 3D geometry modeler but numerical structural analysis and numerical optimization module too, they are suitable for supporting the numerical optimization techniques.

7.2. The main types of optimization tasks, incorporating them into the design process

The simplest task of the engineering optimization is sizing optimization, such as optimization of truss structures, where the topology and the material of the bars are considered as design parameters (Figure 7.1). The continuous design variables are the cross-sectional dimensions of the bars; they can change between his lower and upper limits because of the manufacturability.

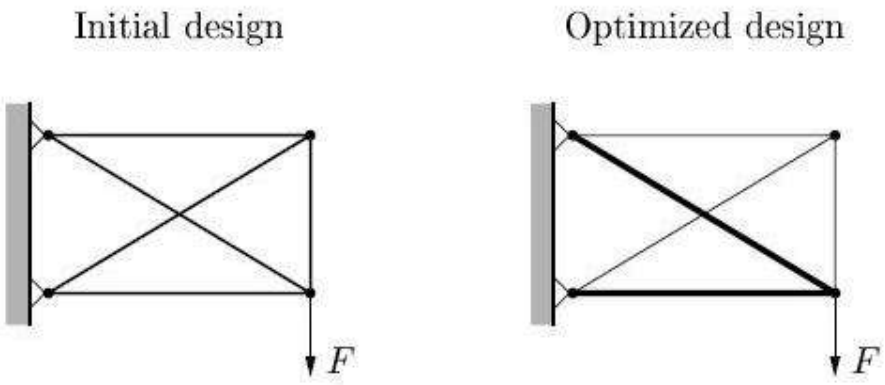

Figure 7.1. ábra: Sizing optimization of the 5 truss bar problem

.

In some cases, partly the surface of the machine part can be varied freely considering the production conditions. This is often happens by casted parts, when the changeable surfaces are not connected to other machine parts. In such cases shape optimization can be used advantageously. Considering a simple supported beam (Figure 7.2), so that the allowable stresses do not exceed the limit. Design variables can only change the bottom contour of the structure and all other properties are set to design parameter. The optimization result (the optimized contour) depends on the number of the design variables. In general, the more design variable means that we can get more information about the optimal shape, but it can lead to numerical instability, which eventually results wavy shape, which is completely useless.

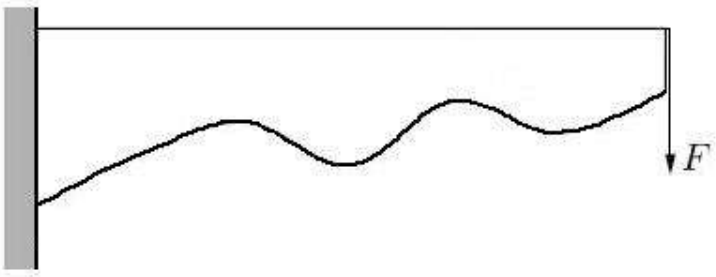

Figure 7.2.: Simply supported beam (starting geometry)

Finally, the most common formulation is the topology optimization. We can apply this technique for optimizing beams and for 2D and 3D solid geometries too. If we allow that the cross sections of the bars become zero, even more bars will disappear in the structure. An example is the design of pillars for high-voltage transmission lines (Fig. 7.3). The figure shows a ruined structure.

Figure 7.3.: Ruined pillar for high-voltage transmission lines

Solving 2D and 3D topology optimization problem first the discretized geometry (Finite element mesh) should be created and the elements separated into two sets: the design elements and the non-design elements. In the non-design region the densities are design parameters and their values either 0 (there is a hole in the structure) or 1 (there should be material). In the design region every element density is coupled with one design variable. After the topology optimization we have a material density for the design domain, which (generally) represents geometry with maximum stiffness (Figure 7.4.). More detailed information about the topology optimization will be presented later.

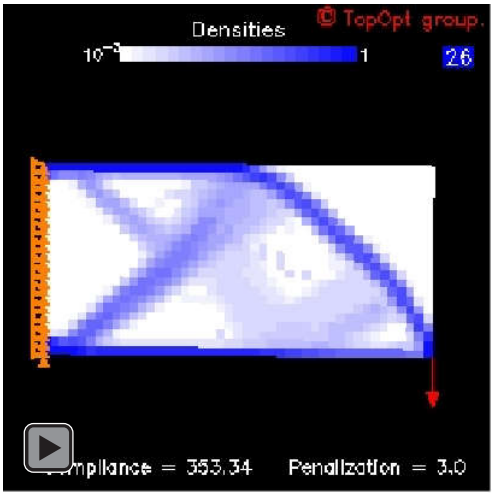

Figure 7.4.: Solving two dimensional topology optimization problems [www.topopt.dtu.dk]

7.3. The optimization examples in the fields of mechanical engineering

Nowdays in the product development process numerical simulation and optimization techniques are shifted into the earlier design stages, in order to coming the products to market earlier, reducing the development costs.
It is also worth to notice, that the number of the different modeles (for example for cars) is much higher than before. It is due to satisfy the changing consumer demands (Figure 7.5.). So the mass production was invented by Henry Ford is now obsolete.

Figure 7.5.: Number of the different models versus time [www.audi.de]

Applying the new CAE systems, with the integrated simulation tools, the designer have the possibility to check the functionality of large number of design ideas parallel. No complicated interfaces and no special simulation tools are needed. Also the number of very costly and time-consuming physical tests can be reduced. (Figure 7.6.).

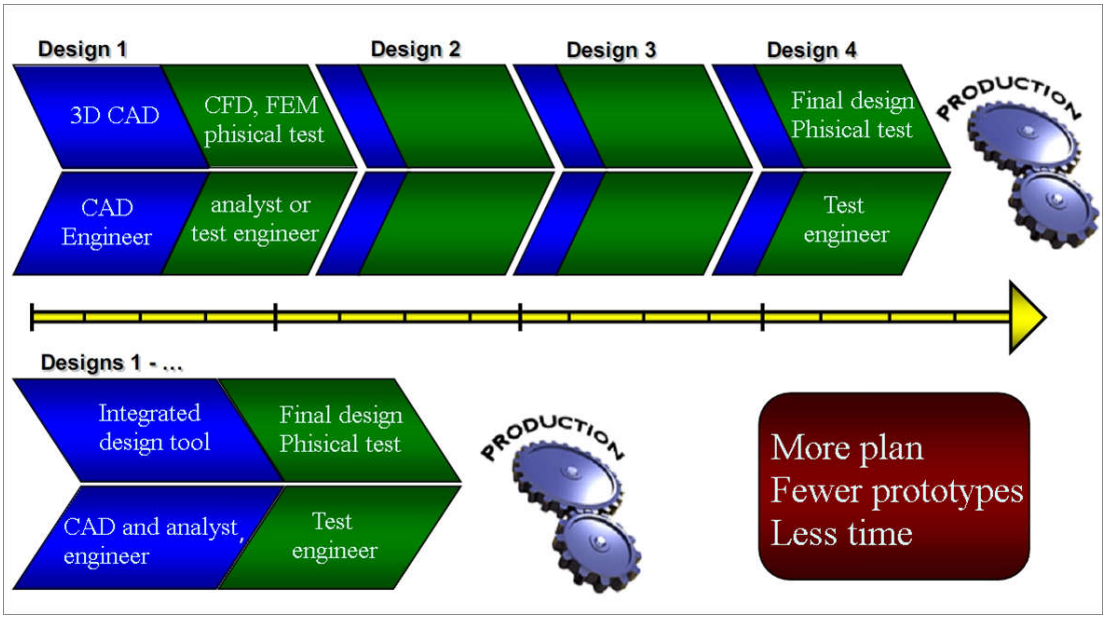

Figure 7.6.: The advantages of using the integrated CAE tools

This technique also allows the designer to use the built in automatic or semi-automatic optimization tools, which increases the probability of finding a better design In this way the structural optimization is linked deeply into the design process.

In the industrial praxis, first the topology optimization techniques are used in the early design stages (Figure 7.7. upper left: defining topology optimization problem showing the design and non-design regions. Figure 7.7. upper right: the result of the topology optimization step). Based on the results of the topology optimization a detailed geometry (CAD model) will be produced by the design engineer which undergoes a shape optimization procedure (Figure 7.7. lower left: the starting geometry for shape optimization, Figure 7.7. lower right: result after shape optimization).

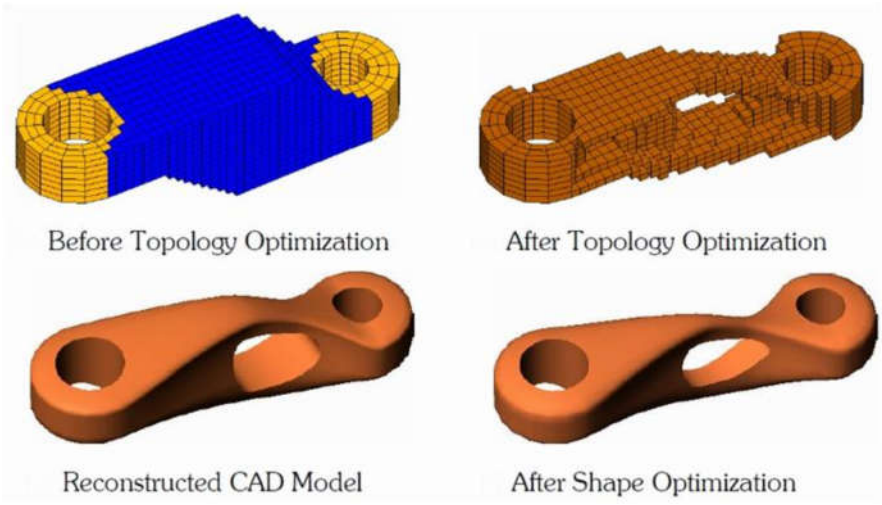

Figure 7.7.: Applying topology and shape optimization of a machine part

Similar optimization application was solved by FE-DESIGN Company using the Tosca system (Figure 7.8.). Using Tosca optimization system large numbers of problems have been solved in the field of automotive industry and also lot of other problems has been solved efficiently.

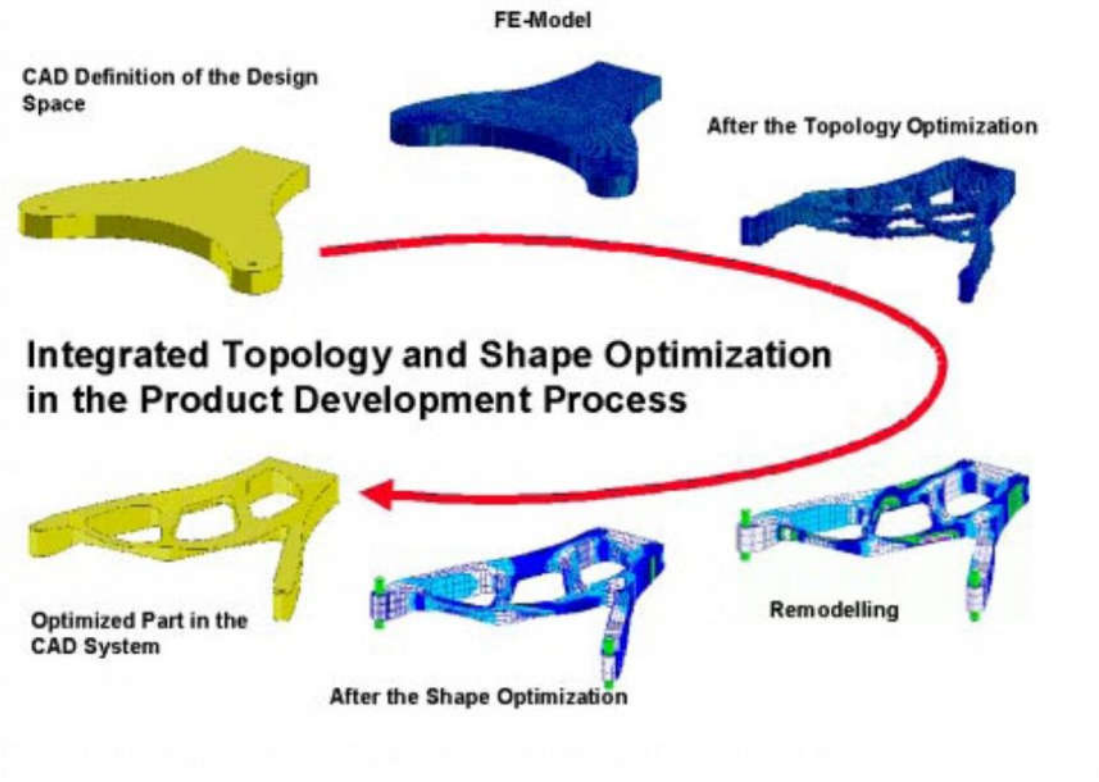

Figure 7.8.: Integration of topology and shape optimization in the product development process with Tosca [FE-DESIGN]

7.4. Questions

1. How can we define a sizing optimization problem?
2 What are the characteristic properties of a sizing optimization problem?
3. What are the characteristic properties of a shape optimization problem?
4. What are the characteristic properties of a topology optimization problem?
5. What is the most relevan design variable in case of solving 2D and 3D topology optimization problem?
6. Why it is important to minimize the “time to market” in the design process?
7. How can we integrate the topology and shape optimization in an industrial design process? What do you think, which approach should be applied in the earlier design stage?

8. GROUPING AND EVALUATION OF THE METHODS FOR SOLVING ENGINEERING OPTIMIZATION TASKS

Because of the diversity of the challenges in engineering, a universal solutions technique cannot be applied. It may be difficult for the reader to find the right optimization method for the problem to be solved. Therefore, in this chapter we will try to review and categorize the presented methods based on the classification presented in the previous chapters, but from the perspective of the tasks to be solved. In some cases, additional methods or examples will be presented too.

8.1. Optimization tasks containing only geometric conditions

We can find such optimization tasks during activities of engineering or product design professionals, where the optimization criteria can also be linked to purely geometric quantities. In this case it's not necessary to solve these computationally intensive real physical processes (for example the numerical solution of partial differential equations).
An optimization criteria is for example, optimizing the external shape of a bottle (or comparing various shape variants), such as the volume of the bottle has to be a certain given value, or between a given lower and upper limit. An equally important geometrical problem is the design of a packaging with minimum material need in case of a complex-shaped part, but with given three-dimensional geometry.
Otherwise, the approximate solution of a problem can be obtained in a way that the simulation is left out. The uniform placement of a car seat's heating element can be approximately examined without the solution of the transient thermal problem, and the solution of certain optical functions (optimization of a simple beam lamp, examination of an object's shadow) can also be approximated on a geometric basis.
The integrated CAE systems are suitable for solving these kind of tasks, where the optimization criteria can be defined by equations, such as the Pro/Engineer, Catia, or SolidWorks systems.
To illustrate the foregoing, we have created an interactive animation, which demonstrates the geometric optimization task through a rotating shaft weight balancing.

8.2. Solving optimization tasks using heuristic optimization procedure

In a design practice sometimes it is necessary to achieve satisfactory results, even in a relatively short period of time, and in the case of a complicated optimization task. The right result means a more favorable result than the initial construction, even if we cannot prove, that the resulting solution is global or local optimization. Such solving methods are called "quick and dirty" procedures. These procedures have a relatively small mathematical apparatus and a wide range of applications. Their application can handle the continuous and discrete design variable problems, and other such cases, when the objective function the optimization criteria is not continuous (for example the optimization on a discontinuous cost function.)
This group also includes a traditional method, also used to minimize volume, that selects the active from the optimization conditions (should be equal to the number of the design variables), and it results in an equation to determine the optimum value of the design variables. In case the active constraints can't be selected, it's possible that in the optimum some con-

straints will not be satisfied, so when this method is used, the optimal point always has to be placed back into the not selected optimization criteria. If, however, all conditions are met, you might want to evaluate the objective function as well, and if it's better than the best so far, then it can be considered an "optimal" solution. If we cannot examine the combination of all constraints, then it won't be sure, that our solution will be the optimum. The method also won't give the optimum, if in the optimum there are fewer active constraints than the number of design variables. The method can be used more favorably in cases of linear objective function and linear constraints, or to solve small tasks; in other cases, the mathematical programming method is recommended. This method is beneficial, if there are relatively few constraints, or the active can easily be selected. The geometric interpretation of the procedure in terms of design variables is the following: through the equations to be solved determine the hyper-plane intersections of the optimization criteria, some of these are on the edge of the allowable range (in which case we accept the solution if the objective function value is positive), others are out of range (in case we reject this point). The former statement is also based on this, that this procedure does not find the extreme if for example the optimum is inside the permissible range, or on the edge, but not in the intersection of the criteria. But in many cases even if the procedure does not find the optimum, it's able to provide a significantly better result than the initial case.

With the increase in the number of variables it is increasingly difficult to determine the global optimization. It's often beneficial using an intermediate step instead of a solution, so we can manage to reduce the complexity of the task and use the "**simple low limit** 'technique. Using this application we are able to estimate the size of each member in the objective function (typically this is used in case of a cost function task consisting of many members), and the ones whose role is small can be neglected. In this way one can create a lower (or upper) limit function, which extreme can be more easily determined than the objective function. The resulting optimum value substituted back into the original and alternative objective function, and the used approximation error can also be estimated. Through the analysis of the **monotonicity test** it's examined if the function respect of the variables is ascending or descending. During the monotonicity test of the objective function, the restrictive conditions also taken into account, the monotonicity of certain variables is often clearly identifiable. Thereafter, it's sufficient to take the general procedures for the remaining design variables, i.e. the dimension of the problem may be reduced.

Another simple optimum calculation approximation method is the definition of the **partial optimum.** In doing so, we can optimize according to the design variables, while we don't change the value of the other design variables. This way always a design variable task needs to be solved, which can be solved using any of the previously explained line side search procedures. As long as we've performed this task for all design variables, we'll get an approximation of the optimum, but if the calculation options allow, the procedure can be repeated in order to clarify the result.

Optimization procedures based on the basics of the probability calculations, for example the Monte-Carlo procedure (the simplest), which randomly selects a point in the n dimensional rectangular box located in the space of design variables, and the optimization conditions and the objective function must be evaluated here. The method is applicable to treat continuous and discrete design variables, continuous and discontinuous objective function, and optimization conditions. The method fairly easily can be connected to any type of structure analysis program, since the program is known as a „black box". Another advantage is, that the method with a given probability (which is nearing 100% by increasing the selected points) converges to the global optimum. A disadvantage is however, that the objective function and constraints

must be evaluated in many points of the design variables space, therefore it has extensive computational requirements, and however it can be perfectly parallelized on multiprocessor or multicore machines. It should be used in tasks, where the numbers of design variables are small, and where the objective function and the constraints are simple analytic functions of the design variables. The method also applies other optimization procedures, for example in the case of the previously described genetic procedures, and the simulated annealing algorithm can also be considered as the enhancement of the Monte Carlo proceedings.

8.3. Optimality Criteria (OC) method for solving stress concentration problems

It's common to establish criteria in this method, which is met at the optimum level, and that's what we try to achieve. The OC methods are only suitable of solving specific tasks, but at the same time are easily programmable and have quick results. Particularly beneficial for these methods, that in most cases the calculation time doesn't increases with the growth of the design variables, and that it usually doesn't include computationally costly gradient calculations. One example of this is the so-called *Fully Stress Design*, which presumes, that you can reach the optimal solution, if in every element the stress reaches the limit stress value for at least one loading case. Other criteria can include displacement, stability, etc. conditions. The strategy for solving these cases consists of the repeated analysis of the structure, so that at the end of each analysis, the construction is changed by simple rules based on special physical properties. From a variety of OC methods we highlight one, which is probably the widest spread in the industry, and that is the Sauter algorithm, which is part of the TOSCA system.
The Sauter algorithm [1] is based on biological analogy: found out observing in a prevailing wind the growth of trees and branches, that the annual rings are not circularly structured, but that they become thicker where there is greater tension in the tree branch. This way the tree strengthens (adds material) the areas with higher stress, in order to prevent the breakage of tree branches.
This observation provides a basis for development of the geometry modification strategy, perfectly suitable for dismantling stress peaks bound to finite-element net, which modifies the geometry of the structure based on the following algorithm:

- The of node location's changing direction is always perpendicular to the surface, or a predefined direction;
- It's extent is determined by the difference of the limit stress and local stress (where the local stress is higher, material is added to the structure, where smaller, it's taken away):

$$\Delta X_n = S\left|\sigma_{red}^{csp} - \sigma_{ref}\right|^{\kappa} sign\left(\sigma_{red}^{csp} - \sigma_{ref}\right), \tag{8.1}$$

where ΔX_n is the n-th finite element node location change in the required (surface normal or direct set) direction, S, κ constant parameters, σ_{red}^{csp} state of stress generated in a node characterized by reduced stress based on a stress hypothesis, σ_{ref} the required reference stress.

Based on stress generated in a given node, the rate of location change of the node in case of different S, κ parameters is shown by Figure 8.1.

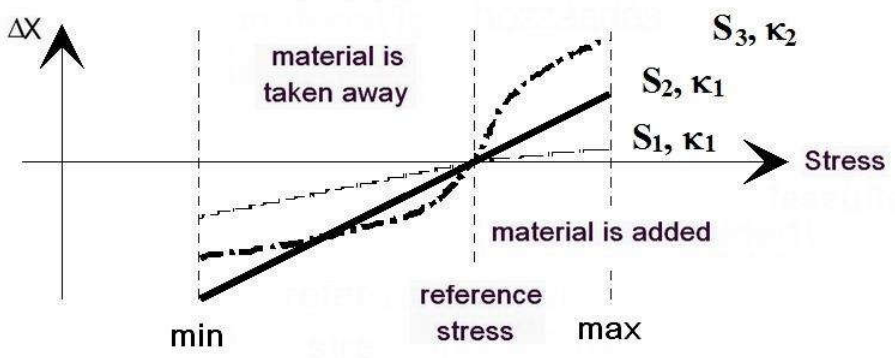

Figure 8.1.: Sauter geometry modification algorithm in case of different constants

When this algorithm is executed in each surface variable node of the examined object, then a new contour is generated. The displacement of the surface nodes leads to deformation of the finite element mesh, that why before any further structural analysis, the mesh needs to be adapted to the new shape, or re-generated.
The efficiency of this method is illustrated on a crank optimization task [8.1], where the test assembly is shown on Figure 8.2. During modeling the bolt pretension, the force from gas pressure, the mass force and the contact relationship between the components should also be considered.

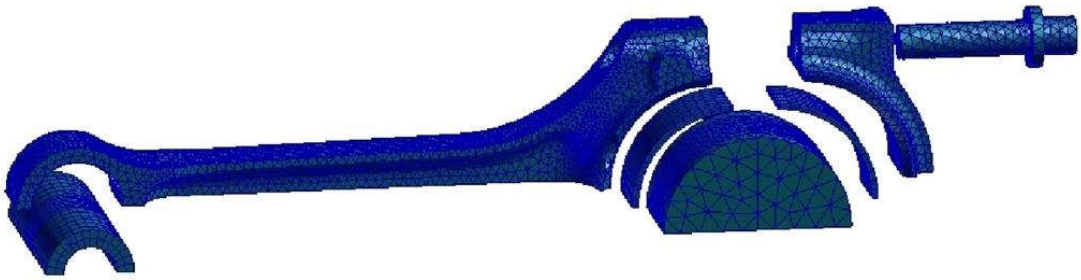

Figure 8.2.: The test rod

The Mises stress distribution on the initial geometry is shown (Figure 8.3.). After five iteration steps, the resulting stress distribution becomes significantly better, and it is maximum value drops to 83% of the original value (Figure 8.4.).

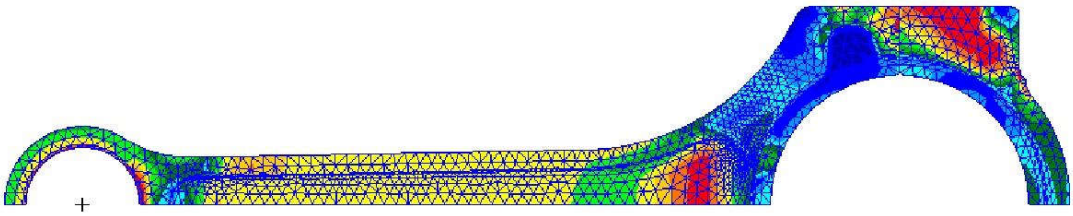

Figure 8.3.: Mises stress distribution in the initial geometry

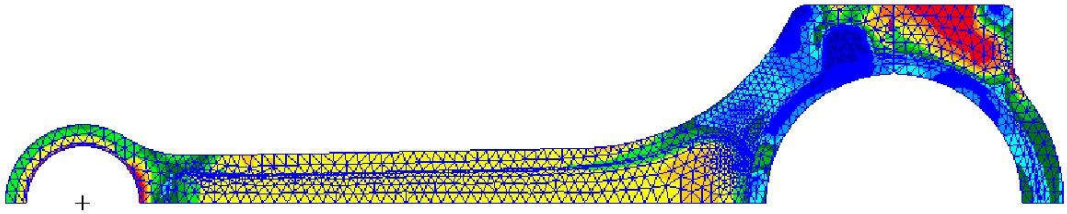

Figure 8.4.: Mises stress distribution on the optimized geometry

The achieved stress decreases during the intermediate steps of the iteration are shown on Figure 8.5.

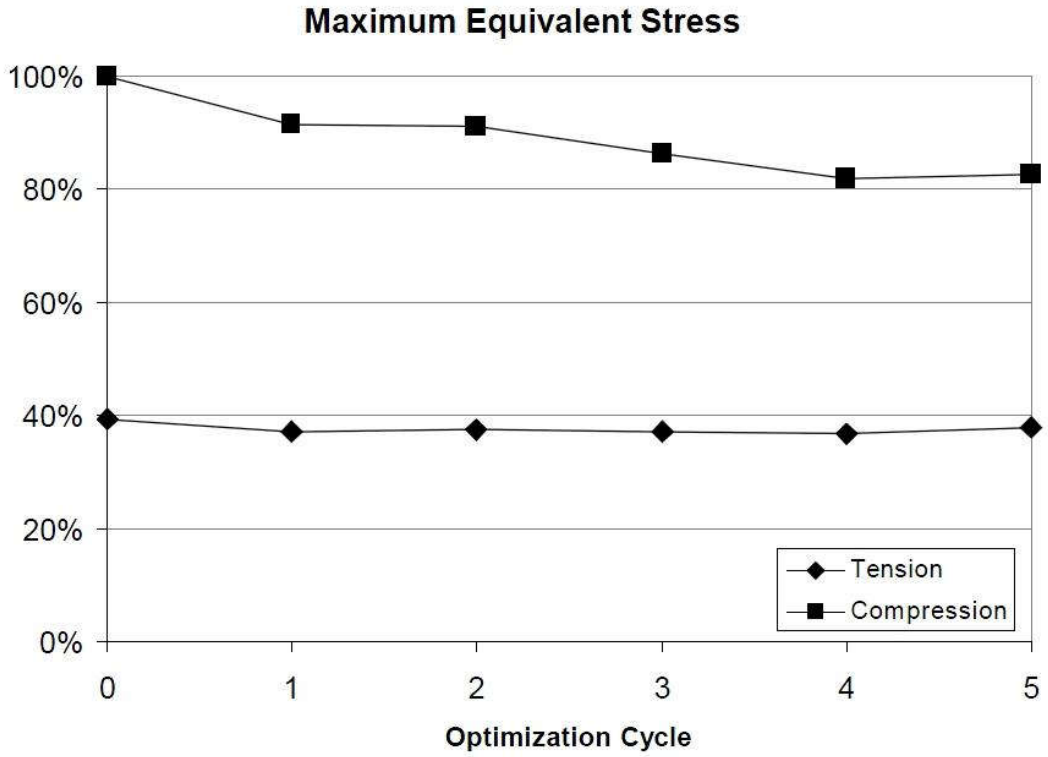

Figure 8.5.: Stress maximum in each optimization step

The advantage of this method is, that approximately during 6 structure analysis CPU time and during a significant peak stress reduction, a new stress distribution was achieved, which is also considerably more favorable in terms of fatigue. However a disadvantage is, that after the finite-element nodes are moved, at the completion of the optimization the changed geometry data is lost and only the surfaces formed by a finite element mesh (for example in STL format) can be exported into a CAD program.

8.4. Mathematical Programming Methods (MP)

The Mathematical Programming Methods (MP) can be widely used for continuous optimization problems. They are commonly used mathematical procedures for the determination of the location of extreme value of multivariable function, also taking into account the constraints (optimization criteria). The simplest such procedure is the Simplex method [2], which can be applied for linear objective function and linear restrictive conditions. Solving non-linear tasks many procedures have been developed, for example the Sequential Linear Programming (SLP), the Sequential Quadratic Programming Procedure (SQP), the Feasible Direction Method, etc. In order to determine the next iteration step, the mathematical programming methods usually require the partial derivatives based on the design variables of the objective function and restrictive conditions. Although capable of solving any problem, at the same time they require quite a large programming effort, and the computational needs are also significant. The MP methods usually guarantee a local extreme value. A great advantage of these procedures is, that they have a mathematically sound shut-down criteria, which is capable of measuring how far away from the optimum is the current iteration step. This convergence criterion is called the Kuhn-Tacker (or Karush-Kuhn-Tacker) criteria.

8.4.1. Lagrange function and the Kuhn-Tucker's criteria

For simplicity consider the optimization task based on the equality of ancillary:

$$Z = F(X) \rightarrow \min$$

$$h_j(X) = 0 \quad j = 1,\ldots,k \tag{8.2}$$

Let's deifne the task's so called Lagrange function:

$$\Phi(X,\lambda) = F + \sum_{j=1}^{k} \lambda_j h_j \,, \tag{8.3}$$

where λ_j is called the Lagrange's multiplier. Necessary condition of the extreme:

$$\begin{aligned} &\frac{\partial \Phi}{\partial X_i} = 0, \quad i = 1,\ldots,n \;\Rightarrow\; \nabla F = -\sum_{j=1}^{k} \lambda_j \nabla h_j \\ &\frac{\partial \Phi}{\partial \lambda_j} = 0, \quad j = 1,\ldots,k \;\Rightarrow\; h_j(X) = 0,\; j = 1,\ldots,k \end{aligned} \tag{8.4}$$

The meaning of the equations 8.5 can be explained that the gradient of the objective function in the optimum can be written as the linear combination of ancillary gradients.
The majority of optimization tasks can be rather formulated by inequality constraints (for example the displacement or stress to remain below a maximum value). Terms of inequality usually labeled with g_j can be reformulated with the introduction of auxiliary variables into equality terms, and similarly the equations for the optimum conditions can be written.
This corresponds to, that X can be a minimum point, as long as in the test point all $g_j \le 0$ constraints are met, and there is $\lambda_j \ge 0$, so that

$$\nabla F + \sum_{j=1}^{J} \lambda_j \nabla g_j = 0, \tag{8.5}$$

where $j=1,...,J$ means active constraints.
This is the so called Kuhn-Tucker criteria, which is a necessary condition for local minimum. It's also a sufficient criterion in case of convex programming problems.

8.5. Comparing the different methods

If a complicated task, within a short time has to be solved in a way, that it's enough to propose a better construct than the starting point, a heuristic procedure can be applied. In case of certain tasks, where the derivative is not available, but a gradient-free OC procedure can be well adapted to the task that should be strictly applied, since it's the most effective way. If the derivatives can be calculated, either analytically or semi-analytically or at least can be approximated by a finite differential, then the gradient methods are effective. In many cases because of the large computing time of the gradient methods, a pre-optimization is performed by an OC procedure (or other suitable technique), from its results the mathematical programming method for optimization can be started. By the application of the defined good starting point, the MP algorithm can find the optimum in a few iteration steps. If it's likely, that the found optimum is only local, than the calculation should be repeated starting from more (or randomly selected) starting points. In case of real, large-scale industrial tasks second-order methods using derivatives are rarely used, because although they are rapidly converging, but the calculation of the second-order derivatives is very CPU time intensive.

8.6. References

[8.1] Meske R., Mulfinger F., Warmuth O.: Topology and Shape Optimization of Components and Systems with Contact Boundary Conditions NAFEMS Seminar: Modelling of Assemblies and Joints for FE Analyses April 24 - 25, 2002 Wiesbaden, Germany

[8.2] Haftka, R.T., Kamat, M.P.: *Elements of structural optimization.* Martinus Nijhoff Publishers, 1985

8.7. Questions

1. Give an example for geometric optimization task.
2. What are the characteristics of "quick and dirty" procedures? Give an example of a procedure, that was divided into this group.
3. What are the characteristic of the optimum criterion procedures?

4. Give an example of an optimum criterion procedure!
5. In case of what tasks can the optimum criteria methods be used advantageously?
6. What are the characteristics of the mathematical programming procedures?
7. What is the Kuhn-Tucker criterion used for?

9. THE ROLE AND METHOD OF SENSITIVITY ANALYSIS; CASE STUDY

Sensitivity analysis has three fundamental applications in solving optimization problems in mechanical engineering:

- Certain optimization methods (e.g. gradient methods) mentioned earlier utilize gradient vectors for defining the optimal direction of the iteration
- In contour optimization an import question is what to choose as a design variable. If we choose too many dimensions, then the task will be unmanageable, on the other hand, if too few dimensions are selected then we only stand a chance to find a partial optimum. In the case study presented at the end of the subchapter we demonstrate how local sensitivity analysis can help to decide if the impact of certain design variables rather designates them to be used as design parameters.
- Sensitivity values also give useful guidance if optimization is performed manually, or if a heuristic approach is applied for geometry selection.

Sensitivity calculation is considered as the definition/obtainment of the objective function and its partial derivative functions along its optimality design variables. There is a multitude of methods to obtain these. If the above functions can be given in a closed analytic form, then they can be calculated either manually or using a mathematical programme package (Derive, Wolfram Mathematica, etc.) and evaluated using the actual values of the design variables. This is nevertheless rare in mechanical engineering applications. If these functions are not available in a closed form and if we can modify the source code of the software used for structural analysis then in most of the cases a semi-analytic or numerical calculus is recommended. Basically, as we move from analytic functions to numerical calculus we get less and less precise solutions (resulting in more iteration steps when for example searching for optimal direction), but in these cases we can obtain a general method which can be simply programmed as an external black box to be used for any structural analysis code.
In the case of finite differences based sensitivity analysis first the structural analysis has to be performed for the given $(x_1,x_2,...,x_n)$ design space coordinates. This analysis can be for example a FEM analysis of linear statics, Eigenfrequency analysis, non-linear statics, thermal dissipation or hydrodynamic analysis or the combination of these. In the next step a sufficiently small Δx value has to be chosen with to which we perturbate all of the n coordinates of the design variable in order to get n new design variables to run the structural analysis on. Before using such a technique it is recommended to assure that the design variables are in a similar order of magnitude.
If these results are ready, then the following relationships can define sensitivities of the objective function and the optimality conditions:

$$\frac{\partial F(x_1,x_2,...,x_n)}{\partial x_i} \approx \frac{\Delta F(x_1,x_2,...,x_n)}{\Delta x_i} = \frac{F(x_1,x_2,...,(x_i+\Delta x),...,x_n)-F(x_1,x_2,...,x_i,x_n)}{\Delta x_i}$$
$$\frac{\partial g(x_1,x_2,...,x_n)}{\partial x_i} \approx \frac{\Delta g(x_1,x_2,...,x_n)}{\Delta x_i} = \frac{g(x_1,x_2,...,(x_i+\Delta x),...,x_n)-g(x_1,x_2,...,x_i,x_n)}{\Delta x_i} \quad (9.1)$$

We note here that this means that for any analyzed point of location *n+1* structural analyses have to be undertaken, which can be rather time consuming. Nonetheless, we might be aware that industrial optimization tasks are usually solved by super-computing tools. Quite advantageously, developments in microchips resulted in today's state-of-the-art CPU-s being multi-core, multi-thread units, providing a good capability for symmetric, parallel task solutions on an average PC's processor. A further option is to set up a cluster of such PCs which can provide a quite strong means of solving optimization problems. Thus, numerical sensitivity analysis (being the most time consuming task of the optimization process) can be performed efficiently. Then, the *n+1* structural analyses is fully independent and can be solved in the run-time of one structural analysis (assuming the availability of the necessary number of CPU cores).

9.1. The direct and adjoint techniques of sensitivity analysis

Two methods of numerical calculation based sensitivity analysis are common: direct computing or computing with the application of adjoint variables. Solution of the equation is based on the finite element technique (linear flexibility statical calculation) and as the fundamental structural response, the optimality condition for displacement is considered. Nevertheless, the procedure presented here can be generalized for other tasks, which can be traced back/originated in the solution of system of linear equations.

9.1.1. Direct sensitivity calculation

The simulation of a linear static finite element is equivalent to the solution of the following system of linear equations, where **K** is the stiffness matrix of the system (a quadratic symmetrical matrix, its order equaling the degree of freedom of the examined problem – for further details see Finite Element Analysis subject), **u** is the displacement vector of the nodes and **f** is the right-side vector computable from stresses:

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{9.2}$$

In direct sensitivity calculation this base equation is derived by the design variables (**X**)

$$\frac{\partial \mathbf{K}}{\partial X}\mathbf{u} + \mathbf{K}\frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{f}}{\partial X}\ , \tag{9.3}$$

which after reordering gives the following relationship for displacement sensitivities:

$$\mathbf{K}\frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{f}}{\partial X} - \frac{\partial \mathbf{K}}{\partial X}\mathbf{u}\,. \tag{9.4}$$

Note, that this is a similar equation to the one which was solved in the structural analysis problem, only the right side is different. This equation has to be solved for all design variables, which might result in a long computation time in case of multiple design variables.

9.1.2. Adjoint method

The optimality condition can be simply identified from the fundamental result of the structural analysis (in practice **q** is frequently a very simple vector, e.g. almost all but one elements equals zero, non-zero on that node or element where we are interested in the dislocation or

stress-sensitivity, and in most cases is independent from the design variables, therefore its partial derivatives with respect to the design variables is zero)

$$g = \mathbf{q}^T \mathbf{u} \tag{9.5}$$

Thus, its sensitivity can be easily concluded from differentiating the above equation:

$$\frac{\partial g}{\partial X} = \frac{\partial \mathbf{q}^T}{\partial X}\mathbf{u} + \mathbf{q}^T \frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{q}^T}{\partial X}\mathbf{u} + \mathbf{q}^T \mathbf{K}^{-1}\left[\frac{\partial \mathbf{f}}{\partial X} - \frac{\partial \mathbf{K}}{\partial X}\mathbf{u}\right] \tag{9.6}$$

With the calculated displacement sensitivities we can quickly define sensitivities of even large number of optimality conditions
If we have a large number of design variables it is recommended to introduce adjoint variables (**a**) according to the following equation:

$$\mathbf{K}\mathbf{a} = \mathbf{q}\,, \tag{9.7}$$

then after a short deduction the following relationship is gained for defining the sensitivity of an optimality condition using adjoint variables:

$$\frac{\partial g}{\partial X} = \frac{\partial \mathbf{q}^T}{\partial X}\mathbf{u} + \mathbf{q}^T \frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{q}^T}{\partial X}\mathbf{u} + \mathbf{a}^T \mathbf{K}\frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{q}^T}{\partial X}\mathbf{u} + \mathbf{a}^T\left[\frac{\partial \mathbf{f}}{\partial X} - \frac{\partial \mathbf{K}}{\partial X}\mathbf{u}\right]. \tag{9.8}$$

Contrary to the direct method, the number of equation systems to be solved equal the number of optimality conditions, and after solving a system of equations the sensitivity of the optimality conditions for all design variables can be obtained using the above equation.
Comparing the two methods, it can be stated that for size and contour optimization problems the direct method is preferred, as the number of design variables is relatively small (e.g. in the range of 5 to 50), but the number of optimality conditions can be large. If we consider a problem of 10 cases of stress with 100,000 elements, where for all elements of the structure a tension criterion is given, then this results in 1,000,000 optimality conditions. If the problem can be formulated with a small number of conditions, then the method of adjoint variables can also be used.
When a topological optimization problem has to be solved with 1-3 design variables per element, thus resulting in 10,000-10,000,000 design variables for the optimization then almost exclusively the adjoint method is used.

9.2. Sensitivity analysis of crank arm– main steps

In this subchapter a component's sensitivity analysis is performed in order to identify which size variable of the potential design variables have the largest impact on the objective function and on the optimality condition. For the analysis any parametric 3D-design framework is capable, which has a built in linear flexibility finite element module (most of the mid-range and high-end CAE systems are such, this being the reason of the selection of this task for demonstration). The following problem is presented by using SolidWorks.
Step 1.: 3D parametric geometry development of the component or composition
Here we don't go through the geometrical design steps as it is relatively simple to develop the examined geometry, and the actual steps depend on the used CAD system (and the background is supported by another educational material). It is important to note nevertheless that

design variables (at least in the first approach) can be selected from the CAD geometry parameters, therefore the development of the geometry (the form feature tree) is very important. The very same physical body can be created with different form feature trees (different order of features), and not all of these are applicable for sensitivity analyses and optimization. There is no generally valid principle for choosing the most proper feature tree, but maintaining a few rules and having adequate experience can aid the choice (unfortunately it is not true that the feature tree used in manufacturing is eligible here, but a tree containing relatively few features can be favorable). It is recommended to consider in advance (at least roughly) which design variables to use and give the initial sizes explicitly. When building the form features the drafts should be fully determined (some frameworks allow for under-determined drafts) since alternations can be better handled this way.
The sensitivity analysis is demonstrated on the example of a crank arm (Figure 9.1 left side), which is a component of sub-assembly (Figure 9.1 right side).

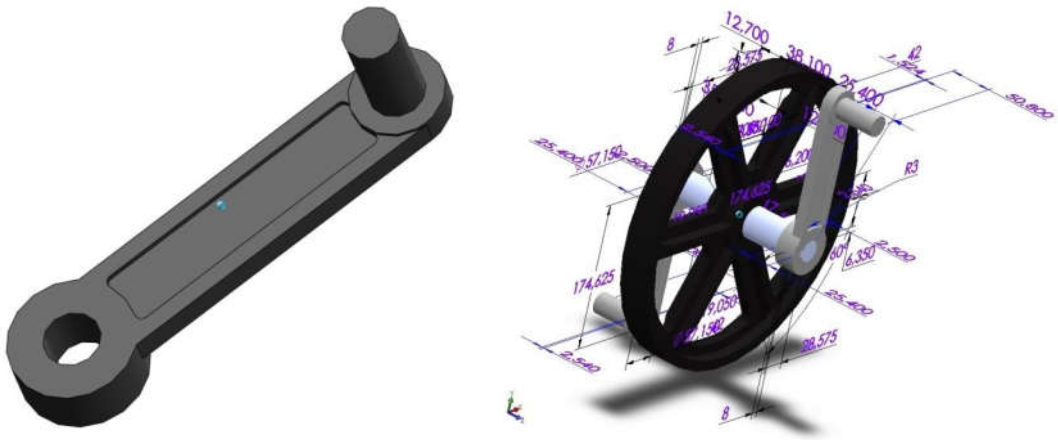

Figure 9.1.: Crank arm and its installation environment

The optimization problem is volume minimization under displacement and Mises-yield criterion optimality conditions. Design parameters are the material characteristics (usual steel characteristics are enumerated), the connector sizes, but beside these still many other sizes can be found on the component (Figure 9.2).

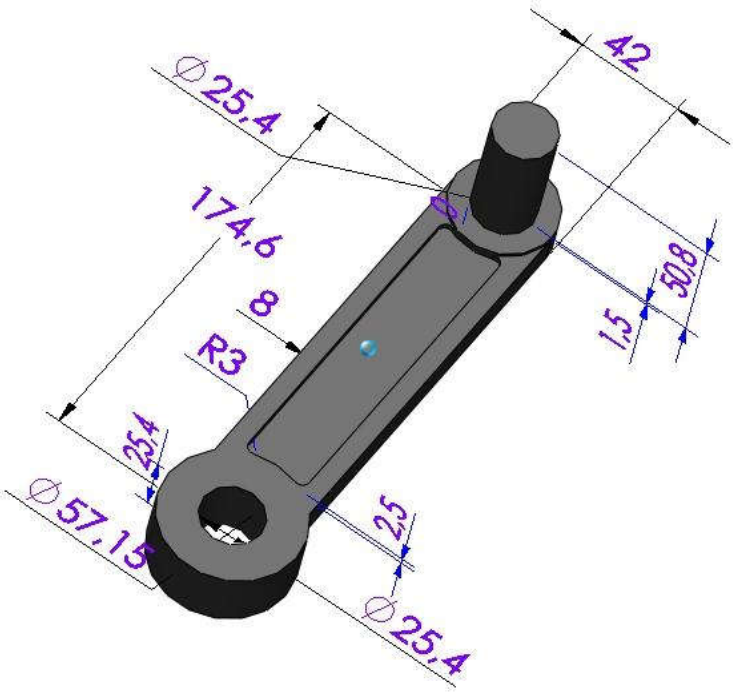

Figure 9.2.: Crank arm and driveshaft size grid

Step 2. Development of a structural analysis model
In the present problem, a linear flexibility statical analysis is performed considering small displacements. The crank arm and the driveshaft are considered as a unit. In our model the arm is fixed to the shaft's axle by rigid coupling, on the driveshaft we assume 1000N of uniformly distributed stress perpendicular to a given plane (Figure 9.3.)

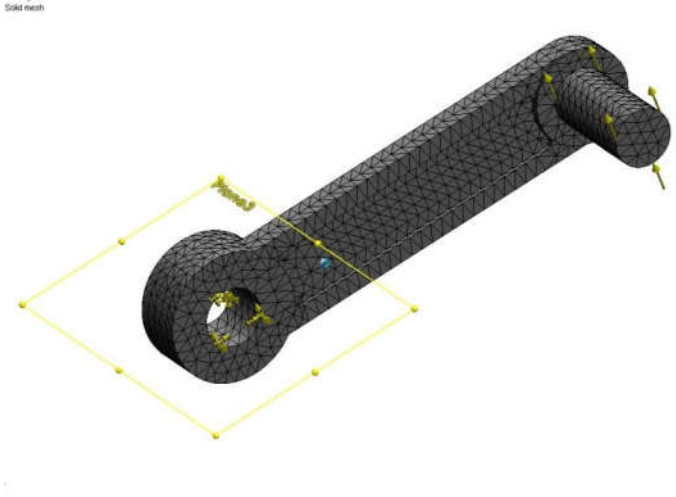

Figure 9.3.: Boundary condition applied at the connecting rod's structural analysis

For the finite element grid an automatic grid generation technique was used utilising second order tetrahedron elements, which is also recommended for evaluation of stress type conditions (if only displacement-type optimality conditions are given, then linear elements can be considered).
The evaluation of the objective function can be realised by a CAD program (in this case 151,88 g), however, for the evaluation of the tension optimality conditions a finite element calculation is necessary, the results of which are shown on Figure 9.4. It is visible that the maximum of tension is at the stem of the hub (110 MPa). In case of a tension optimization condition special attention has to be paid that the necessary level of accuracy in stress calculation is maintained, since the calculation is based on these values. In such cases a convergence analysis should be performed by further refining the finite element grid. In this case it is sufficient to perform a local grid refining in the high tension zone. It can be seen also in this case that the maximal von Mises tensile stress arising in the component grew up to 136 MPa and on the surfaces containing the critical rounding the maximal value grew to 123 MPa. It can be therefore stated that the precise structural analysis model's development is inevitable when preparing for a sensitivity analysis.

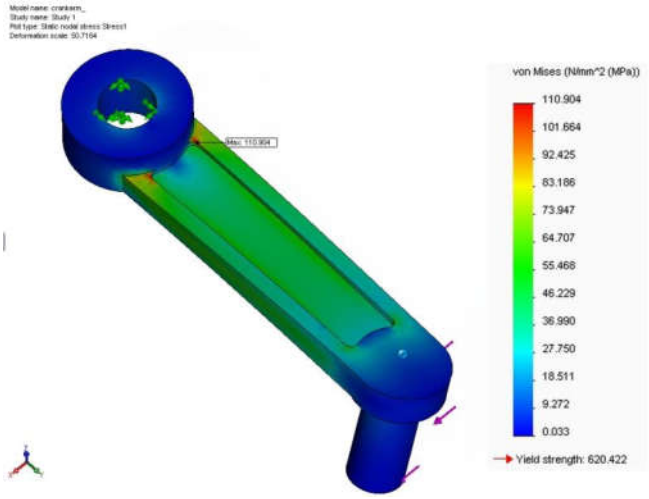

Figure 9.4.: Tensions arising in the crank arm– initial model

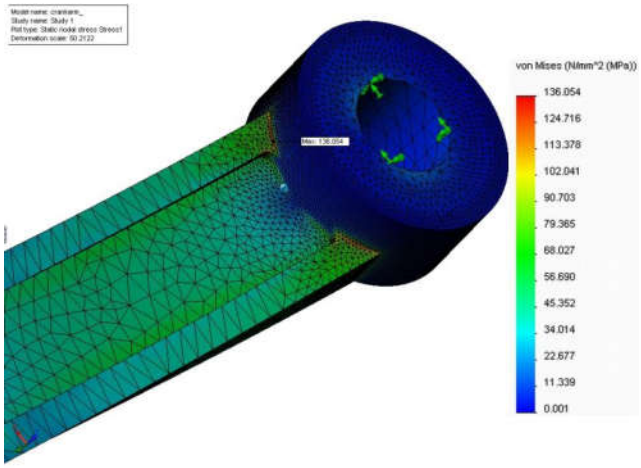

Figure 9.5.: Tensions arising in the crank arm – refined model

Step 3. Selection of design variables
Based on the results the designer formulates an idea which sizes to change on the component in order to save on material requirements while maintaining the required carrying or stress capacity. To avoid too much calculations 4 size variables were defined (Figure 9.6.): width of connecting rod (initial size: 42mm), radius of rounding at alleviation (initial size 2.5mm), considered to be identical on both sides, and the stub size remaining at the edge of alleviation (initial size: 8mm).

Table 9.1: The design variables and their initial values

No. of design variable	Varying size	Initial value[mm]
DV 1.	Width of connecting rod	42
DV 2.	Radius of rounding at alleviation	R3
DV 3.	Depth of alleviations	2,5
DV 4.	Stub size remaining at the edge of alleviation	8

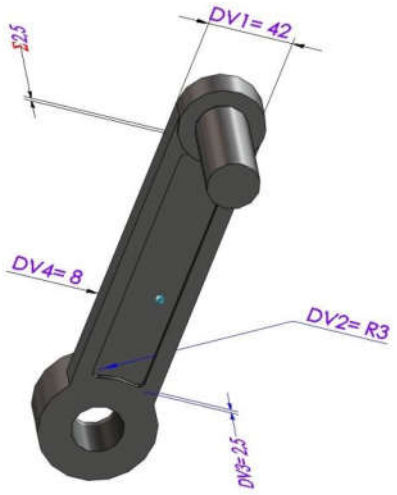

Figure 9.6.: The selected design variables

The sensitivity analysis of the objective function and the optimality criterions (maximal Mises stress on the surface containing the critical rounding) was performed for all four design variables with the presented finite difference technique.
Before turning our attention to the actual problem's results, let's consider some general remarks helping us in the solution.
It is always advisable to assess if the achieved sensitivity results are of usable accuracy, otherwise no decisions can be based on these. Most leading CAE systems, directly or indirectly have such a tool, which help defining sensitivity values. For example, in Pro/Engineer there is a direct option for defining and displaying sensitivity values on a 2D-chart. On the other hand, in SolidWorks a so called „design study" can be defined to calculate sensitivity the easiest way, design variables can be easily given, the objective function and the optimality criterion is done in the geometrical model as sensor.
In solving the task the latter software was used to demonstrate the impact of the respective Δx perturbation values (a separate design study was defined for all perturbation values).

Results can be simply transferred to Excel, where finite differences were evaluated and the sensitivity values were displayed.
It is especially recommended to pay attention in the sensitivity calculation of stress type optimization criterion. This on one hand is the most sensitive to calculation error; on the other hand the maximal tensile stress arising in the structure can sometimes give misleading optimal results. This can be seen in our example (intentionally) where the maximum tension is shown in the environment of an existing, but non-modeled rounding. The value of this „maximum" does not give reference on the structure there it is not recommended to use in the optimization. In these cases rather choose a surface where the maximum tension is a characterizing information (in mainstream CAE systems it is allowed to deviate from only choosing tension arising in the whole structure). It is more favorable if we want to limit the maximum displacement, its accuracy can more easily be guaranteed. An even more favorable situation if we evaluate a condition or an objective function coupled with some geometrical parameter (for example, in the case of volume as objective function no structural analysis is necessary), in this case the sensitivity values can be gained with high precision and independently from the applied discretization.
After these remarks, let's return to the presentation of the problem and the conclusions. For determining the sensitivity values not the direct size values of the geometrical variable were considered as design variables but their values divided with their base value, thus normalising the variables. This technique is especially preferential when there are differences in magnitudes between the variables. In determining sensitivities the value of Δx was chosen to be 10%, 1% and 0.5% respectively.

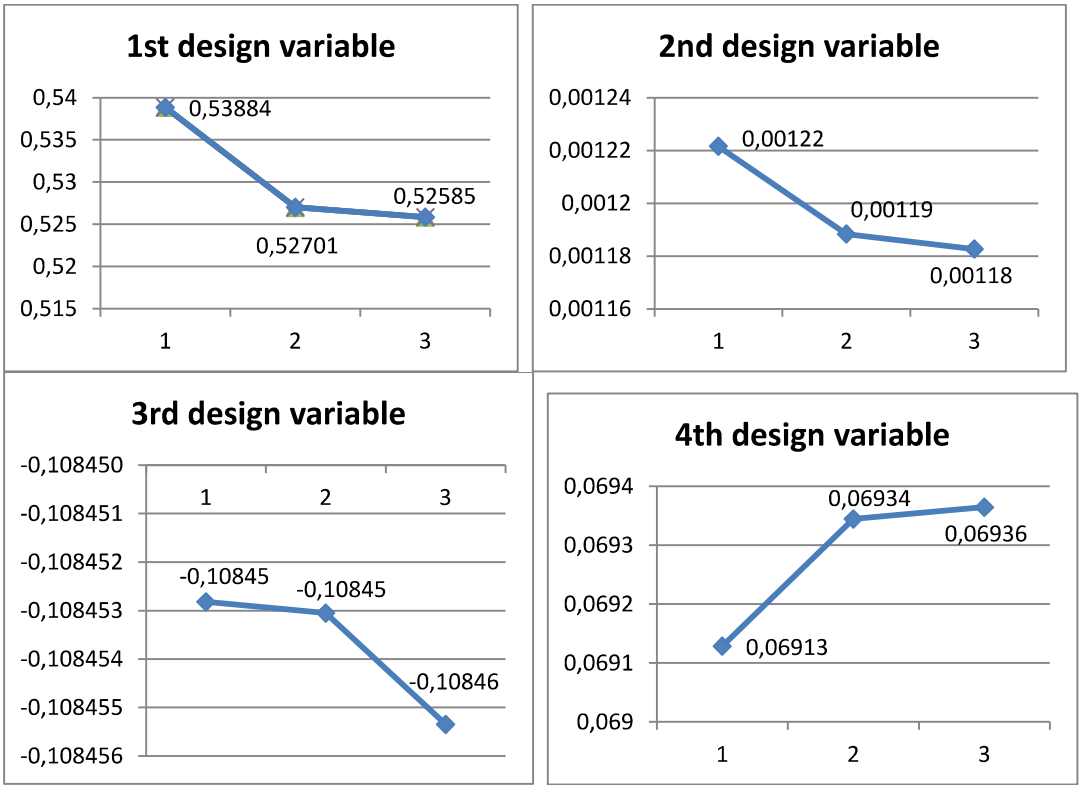

Figure 9.7.: Determination of the sensitivity of the objective function

The determination of the sensitivity of the objective function is relatively easy, and it can be seen (Figure 9.7), that its accuracy hardly depends on the chosen Δx value. The largest impact on volume comes from the first design variable, the width of the crank arm, the smallest impact from the second design variable (radius of rounding). If for example a design variable is chosen which is not connected to the geometry (e.g. the material's modulus of elasticity), then the objective function's sensitivity will be zero.
The reliable determination of sensitivity for stress type optimality conditions is a rather more complex problem. Using an automatic grid generator does not guarantee the identical location of nodes used for the finite element calculation, neither that the structure of the grid in the locality of the analysis remains the same, when we slightly change the geometry and re-grid the whole body. Thus, although the first and third design variable allowed for an easy convergence, but in the second and fourth variable's case an additional smaller (0.1%) point was necessary (Figure 9.8.Figure 9.8.).

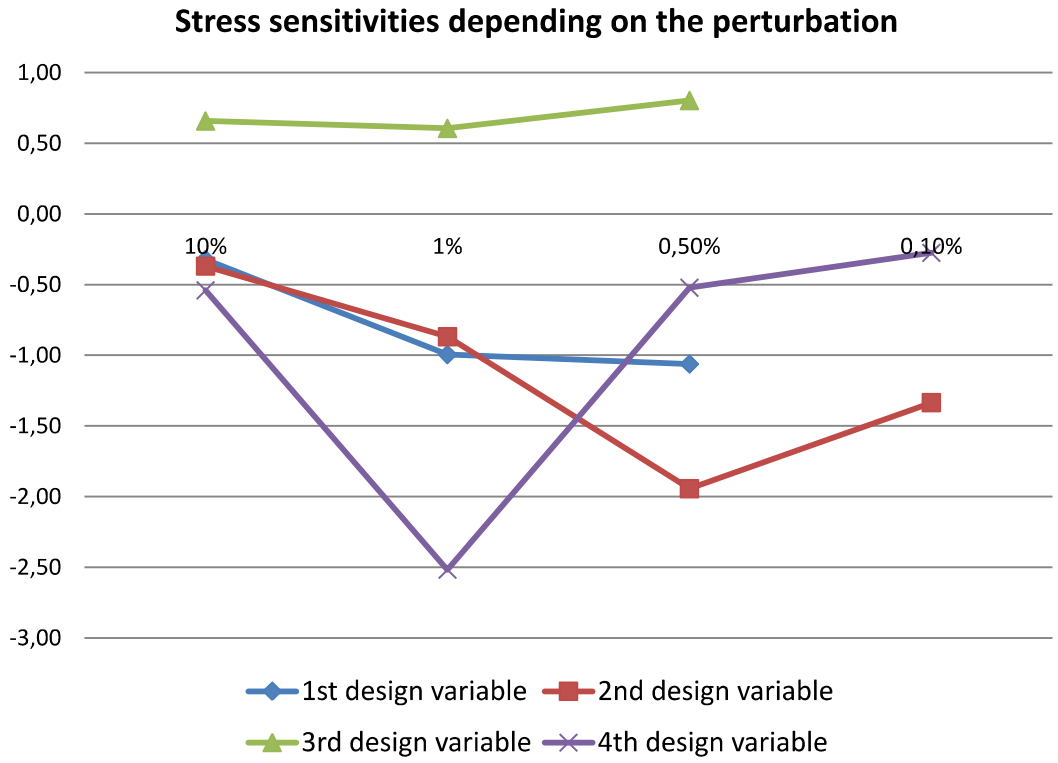

Figure 9.8.: Determination of stress sensitivities

These results, compared with the objective function's sensitivity allow us to draw the conclusion that although the second design variable has the least impact on the objective function but it still has the most significant role in determining maximum tensile stress.
It can be concluded furthermore that changes in all design variables have an impact either on the objective function or on the stress optimality criterion, but if the number of variables would still had to be reduced, then the fourth variable could be omitted, as its sensitivity values are relatively small.

From the engineer's aspect the maximal deformation of a structure can be important, therefore the results for maximal displacement sensitivity calculations are presented (Figure 9.9.) for the sake of completeness. It is well visible from the chart that displacement sensitivity converges more easily than that of tensile stress, thus its determination is easier. According to our expectations, changing the radius of rounding as the second design variable does not have significant impact on maximal displacement.

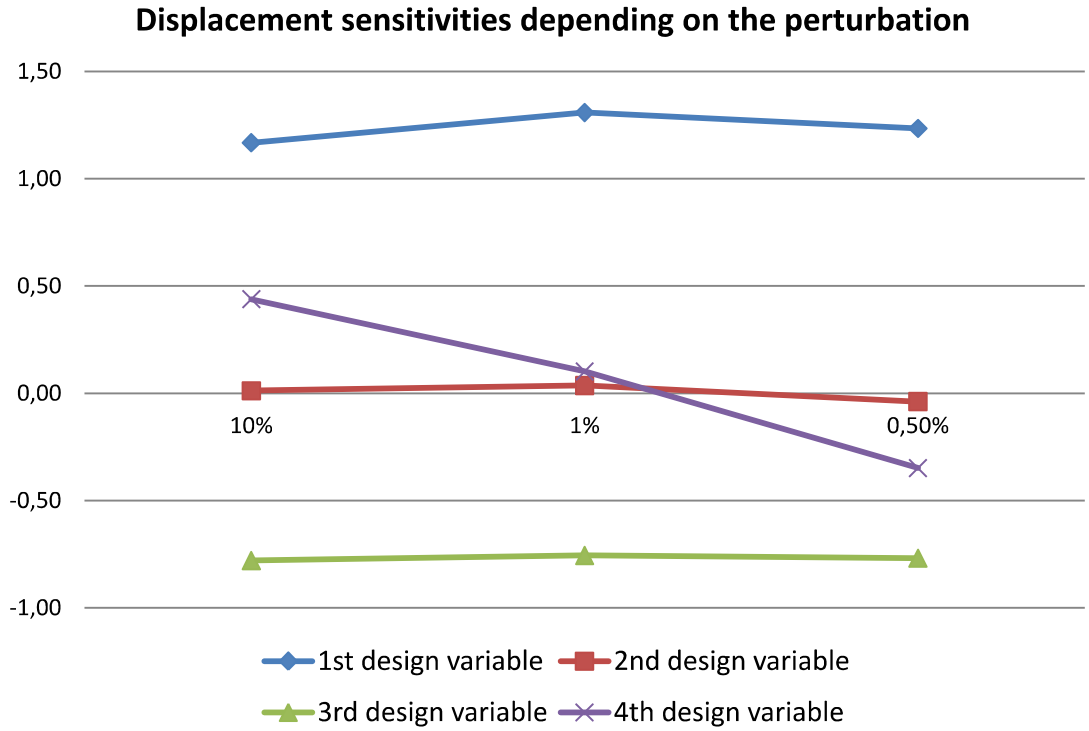

Figure 9.9.: Determination of displacement sensitivity

Summarising the results, in this subchapter we highlighted the importance and applications of sensitivity analysis. We made an overview of the generally applicable finite difference technique, and introduced the fundamental equations and applications of direct and adjoint variables methods. An actual problem was used to demonstrate the method of determining sensitivity values highlighting when and which values can cause numerical stability problems. We showed how to test convergence to ensure the accuracy of sensitivity values and what type of conclusions can be drawn on design variables in such calculations.

9.3. Questions

1. Which areas of applications were mentioned for sensitivity analysis in solving engineering problems?
2. How is finite difference sensitivity calculation technique defined?
3. How does step size influence the precision of finite difference sensitivity calculation?
4. When should direct and when should adjoint sensitivity calculation be rather used?
5. Which are the main steps of sensitivity analysis?

10. SHAPE OPTIMIZATION. GEOMETRIC PARAMETERS AND THEIR IMPACT ON THE OPTIMUM

During shape optimization the data set describing the initial structure is split into two parts: one is a set of design variables, the other is a set of design parameters. During shape optimization the design variables should be selected from the descriptive geometry subset. At the application of parametric 3D CAE systems, the data set describing the geometry consist of the dimensions and order of features, and the definied sketches and specified dimensions and geometric constraints (perpendicular, tangential, pressure, etc...). An examined part can be described by a variety of feature sequence; they will be called geometric representations of the part.

10.1. The effect of the geometry description on the optimization model - a case study

To illustrate the differences based on the variety of describing geometry during optimization, a simple example can be presented: Figure 10.1. shows a rectangular plate with a hole and it's dimenions.

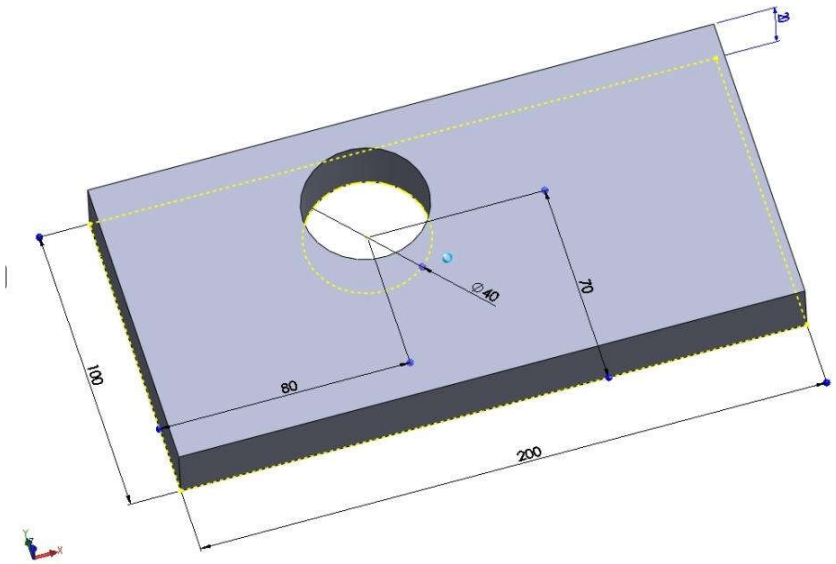

Figure 10.1.: Rectangular plate with a hole - CAD model

Even in the case of this very simple geometry we have multiple opportunities to develop the feature-tree. In the most simple case in which we create the body by one extrude feature from single sketch, containing a rectangle and a circle with given dimensions.
In case the diameter of the hole is increased, the solid geometry disappears when diameter reaching the 60 mms and CAE system is unable to create the body using this simple extrude feature (Figure 10.2. and 1_feature_regen hiba.avi).

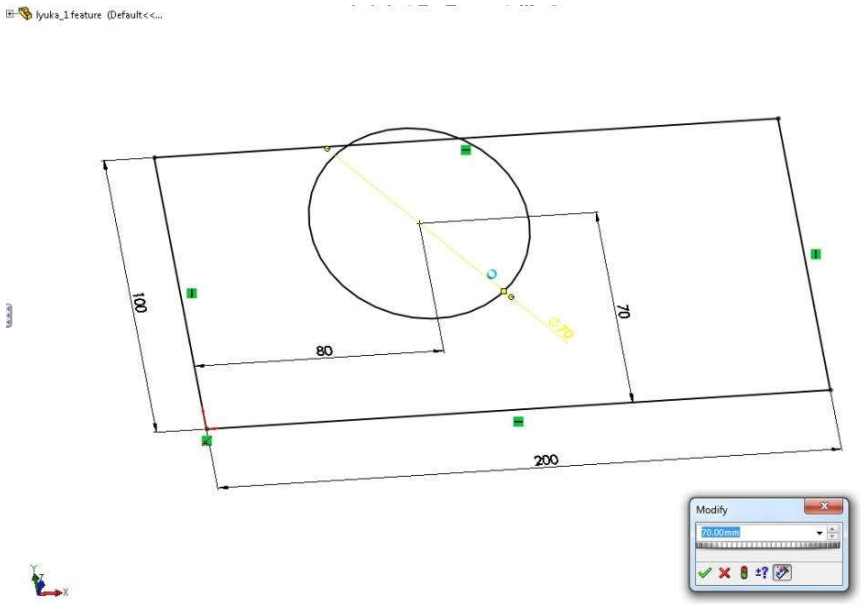

Figure 10.2.: The collapse of the geometry because of the contour intersects

The geometry may also be created by using two features: starting with extraction the rectangular and then drill a hole into (Figure 10.3.). When increasing the size of the drill hole by reaching 60 mm the hole disappears, because the geometric modeler can't construct the cut feature (Figure 10.4. and 2_feature geom .avi).
If we further increase the diameter, the tested CAD system has been able to build up the geometry (Figure 10.5.), but the surface topology of the constructed geometry is different from the initial, which is not recommended in shape optimization. In this case the number of surfaces increased by one. The attached animation (3_FEM changes in boundary conditions.avi) shows, that when on the variable surface finite element boundary condition is given, the FEM modeling in general cannot properly handle the problem.

Therefore, when designing the geometry of the following should be kept in mind:

- the correct structure of the feature tree
- the correct choice of design variables and variation limits
- it should be checked that the structure geometry can be re-build in all of the search range and the surface topology would not change

In spite of the difficulties outlined the advantage of the parametric shape optimization is that parametric surfaces can be easily produced and that the method is easily integrated with CAD systems, as well as the optimized geometry immediately is generated in the usual CAE system.

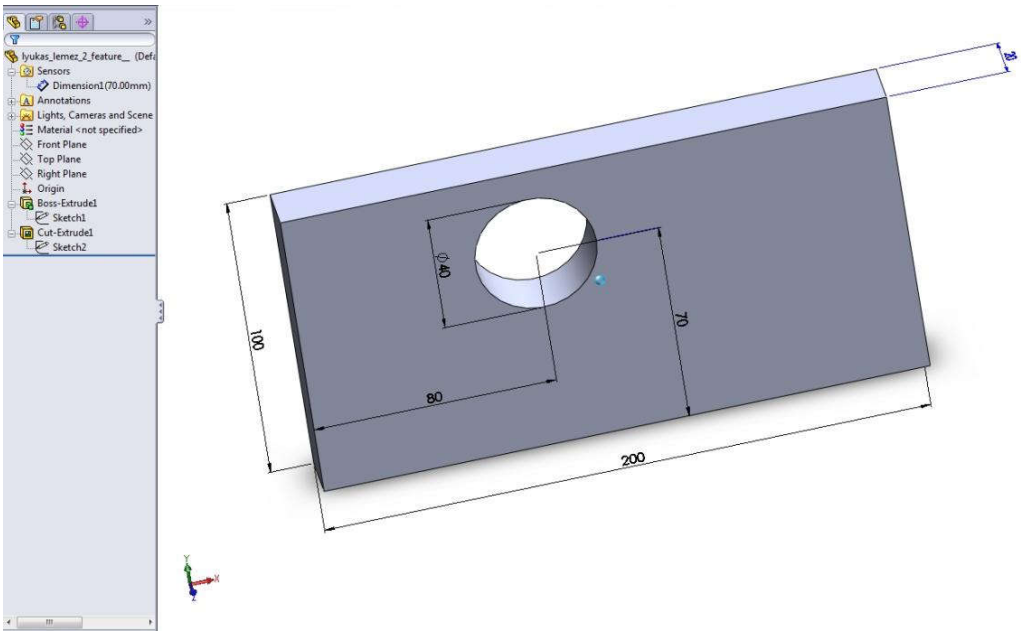

Figure10.3.: The structure of the rectangular with a hole based on two shape features

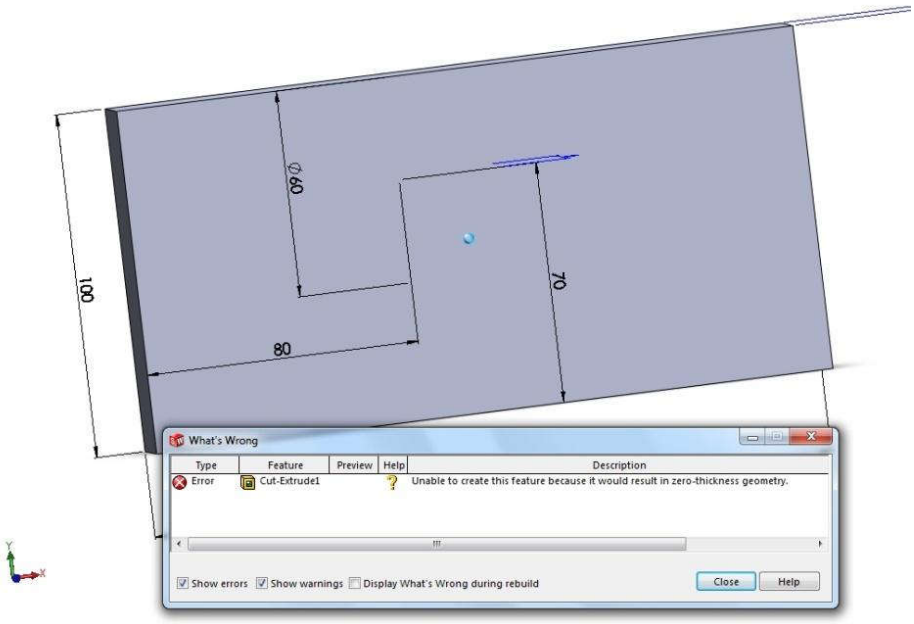

Figure10.4.: The cutout shape feature causes an error, the hole disappears

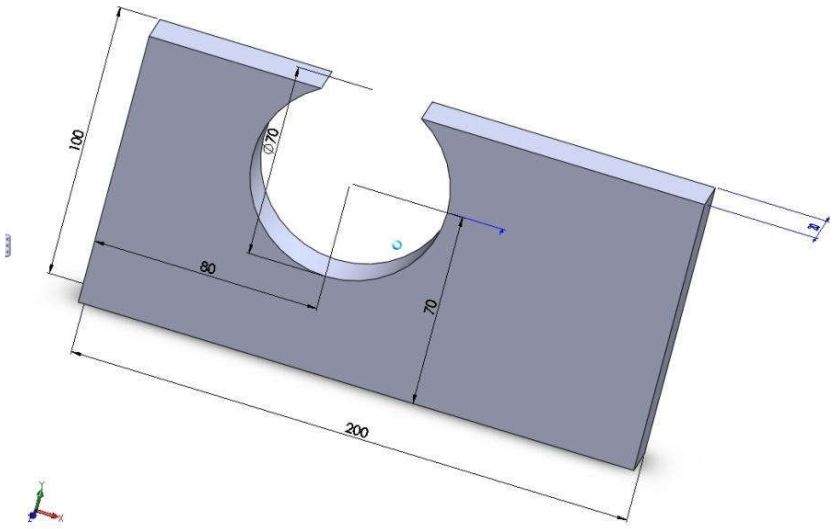

Figure10.5.: Regenerated geometry with the change of the surface number

10.2. Questions:

1. Give examples how the shape feature sequence used in the construction of the body impacts the determination of the optimization range!
2. What kind of problems can cause the changes in topology geometry during reconstruction of the finite element model?
3. Why is it advantages a two-way associative data link between a CAD system and a structural analysis software?
4. What are the most well-known CAE systems, which have both a structural analysis and an optimization module?

11. TOPOLOGY OPTIMIZATION

This chapter shows the place and role of topology optimization methods in the design process, presents the basically different types of methods and demonstrates their benefits and drawbacks. The steps of problem solving will be demonstrated on a simple problem.

11.1. Place of topology optimization in the design process

In a very early stage of the designing we must decide the layout of the designed object for the given operating conditions. In this stage there is possible to achieve great profit on the base of low cost; later on, every small change implies great investment.

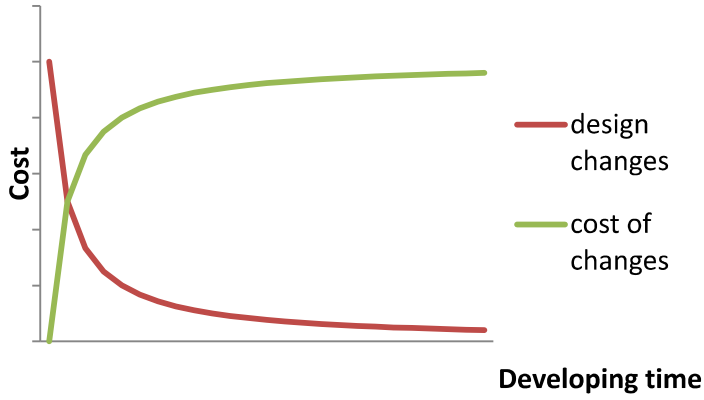

Figure 11.1. Topology optimization can result cost saving in the conceptional phase of the design

On one hand, having a very complicated designing problem, it is worse to recline on the empiric and intuition as in such a case the problem is hard to formalize. On the other hand, the problem can be very new and we are staying there without any specific knowledge about the task, it is worse to search the main effects and we are looking systematically after the best solution. In the case a layout is chosen, we are restricted in the goodness of reachable improvements so we need such a procedure, which supports such a serious decision. The only information about the design is the loading situation, the mathematic model describing the physical behaviour and the connection to the environment. We have no information about the topology of the body at all: how many ribs are needed to enforce the structure o in the contrary how many lightening hole and where we need. The ambition of the topology optimization is to decide which part of the given space should be occupied by our design optimally in a specified view. Nevertheless, the result is rough so it should be further refine having other standpoint, connectivity model and conditions applying in a following size and /or shape optimization loop. Choosing or constructing a topology optimization method the first focus is the characteristics of the problem itself: what the objective(s) is/are; the second one is the way of parameterization to follow the topological changes.
As the evaluation of the solutions follows by solving the physical model through discretization it is important how the parametric representation of design connects to the physical one's.

Those methods which use a mesh on the changing design, have difficulties as the topology changes. For topology optimization it is more suitable to model the geometry on an implicit way and perform the structural analysis on a fixed, initially given domain. These methods can adapt to the topological changes and there is no need for remeshing the changed body.

11.2. Benchmark problems for testing the algorithms

Mitchell laid down the theoretical base of topology optimization by giving optimality criteria for light structures. Later, in the 70'ths with Rozványi and Prager he extended the theory generally and based the examination of continuous structures. These results serve for comparison to the different topology optimization methods [11.1].
The solution of the here demonstrated examples gilt for one load case, with elastic behaviour. The results for weight minimization are valid for both stress and compliance constrain as a relation exist between the optimal weight W_S, the allowable stress σ_0 and the optimal weight W_C for a given compliance (11.1)

$$W_C = \frac{\sigma_0^2}{\rho E C} W_S^2 \tag{11.1}$$

ρ is the specific weight, E the elastic modulus. The relation between the optimal weight and the length of the structure can be given with dimensionless data $\overline{W}$ and $\overline{L}$ (fig. Figure 11.3.), where h is the height of the beam and P is the load.

$$\overline{W} = \frac{W\sigma_0}{Ph\rho}, \overline{L} = \frac{L}{h} \tag{11.2}$$

Mitchell beam – it is fixed in one end and loaded on the free one. The analytical results can be seen on Figure 11.2.

Figure 11.2. Analitical solution of the Mitchell beam

MBB beam
The problem was originally formulated by the air company of Messerschmidt-Böhlkow-Blohm. In the literature the compliance minimization of the simply supported beam is called MBB problem (Figure 11.3.)

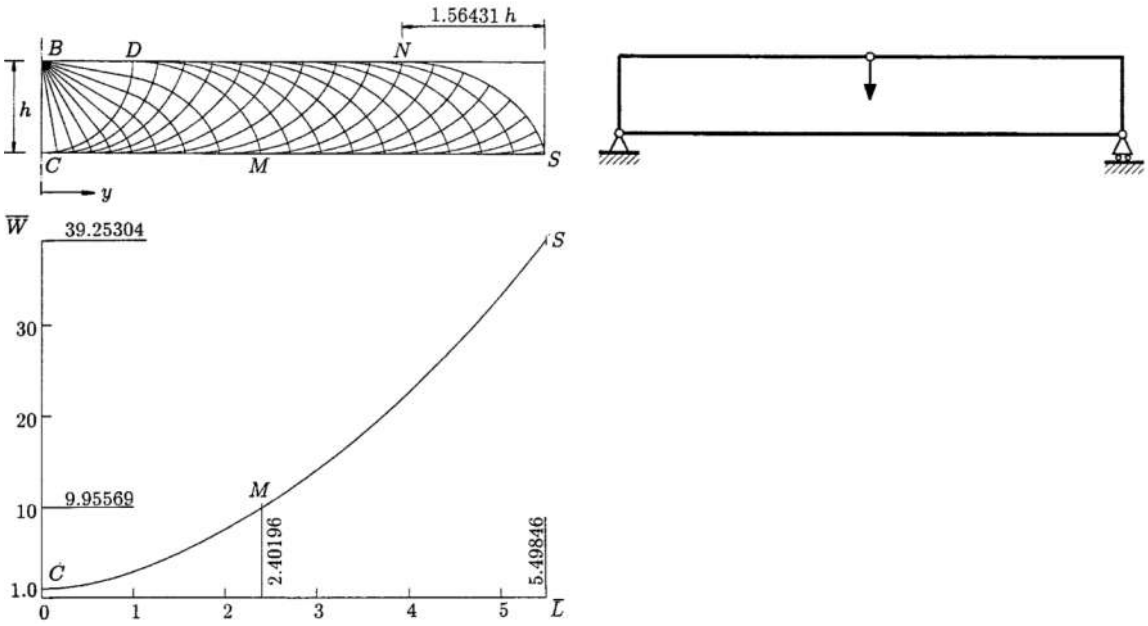

Figure 11.3. Analytical solution of MBB structure

11.3. Methods of topology optimization

In the last decades many topology optimization procedures were developed as the homogenization method, the material distribution method SIMP (Solid Isotropic Microstructure with Penalty), evolutionary techniques (ESO = Evolutionary Structural Optimization), reverse adaptivity, the bubble method, the level set method (LSM), method of topological derivatives and others. The most effective and simple is the SIMP method to solve the basic problem of compliance minimization so it is widely used in the commercial programs and for this reason it will be shown in more detail. Further extension and application can be found in [11.15].

Topology optimization methods can be sorted in three groups:

- methods based on mathematic background which are suitable for global criteria as compliance or volume
  - homogenization method
  - SIMP
  - level set method
  - phase field method
- heuristic methods which stand for homogenization with removing small amount of material from the design space where the governing criteria e.g. stress is low.
  - soft kill (SK) or a hard kill (HK) evolution methods (ESO, AESO, BESO, XESO)
  - reverse adaptivity
  - metamorphic development
- mixed method with combining topology and shape optimization
  - bubble method,
  - isoline method.

In the following sections the basic characteristics of the above methods will be shown.

11.4. Homogenization method [11.2]

By the homogenization method the body is treated to be from a porous material and we are looking for optimal distribution of microscale voids which shape is assumed to be rectangular in a given domain. Applying finite element method, this domain is divided in N finite element and design variables are the size and direction of the voids in each of these elements (Figure 11.4).

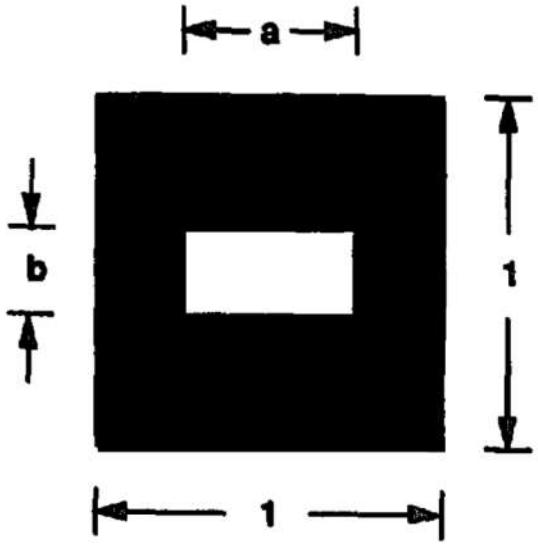

Figure 11.4. Design variables in the case of homogenization method

Rectangular holes are chosen because they can realize the complete void ($a = b = 1$) and solid ($a = b = 0$) as well as generalized porous medium ($0 < a < 1,\ 0 < b < 1$). Having the variables a and b inside of the intervals, the real structure must be porous or inhomogeneous density which is hardly to carry out. Additionally, in this case we have 3 design variable pro elements resulting a very big optimization problem; it can be slightly reduced for example by applying square voids. However, it was the first method applying fix domain for structural analysis and this main idea was further developed by the SIMP method.

11.5. The SIMP method [11.3]

The SIMP method is the most popular topology optimization method due its simplicity. Solving form finding problems of mechanical engineering, the optimization criteria is mostly the stiffness of a part, for one or more load case constraining the volume. The achieved proposal must be interpreted taking manufacturing viewpoints into account. The reformulated geometry should be further examined refining the loading environment, or other conditions can be considered.

The matter of the method is to search the optimal topology of the structural part as material distribution in a given design domain divided into N elements. The design variables are the fictitious ρ_e relative element density, ($e = 1, 2, \cdots, N$). The element e is solid, if $\rho_e = 1$, and void, if $\rho_e = 0$. Avoiding the computational difficulty of the non-continuous description, in place of discrete values $\{0, 1\}$ we use continuous design variable $0 \le \rho_e \le 1$, so

$$E_e = \rho_e^p E_0 \tag{11.3}$$

where p is the penalty power penalizing intermediate densities (usually should be 3 to 5), E_0 is the Young modulus, that means we interpolate with ρ_e^p between 0 and E_0.

Compliance minimization
The topology optimization problem can be given as follows:

we are looking for $\boldsymbol{\rho} = \{\rho_1, \ \rho_2, \ \cdots, \ \rho_N\}$

$min : C = \boldsymbol{f}^T{}_k \boldsymbol{u}_k$,

subject to

$\sum_{e=1}^{N} \rho_e v_e \ \leq V^*$,

and $0 < \rho_{\min} \leq \rho_e \leq 1$ (11.4)

where displacement $\boldsymbol{u}$ depends on the vector of design variables $\boldsymbol{\rho}$, $\boldsymbol{f}$ is the vector of outer force, and $\boldsymbol{u}$ is the solution of

$$\mathbf{K}(\boldsymbol{\rho})\,\mathbf{u} = \mathbf{f}, \tag{11.5}$$

where $\mathbf{K}$ is the global stiffness matrix, composed from the element stiffness matrices which depend on design variable ρ_e threw the above introduced way. V^* is the prescribed volume, v_e is the element volume. The lower limit for density $\rho_{\min}$ was introduced to avoid singularity. Index k denotes the degrees of freedom where the outer load acts.
As the work of the outer load is equal to the energy stored in the deformed body, the objective function can be given as follows:

$$\min\ C = \mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u}, \tag{11.6}$$

that is our problem is to minimize the strain energy.

Introducing the Lagrange function the necessary condition of optimality

$$\mathrm{p}\rho(\mathrm{x})^{\mathrm{p}-1}\mathrm{E}^0_{\mathrm{ijkl}}\varepsilon_{\mathrm{ij}}(\mathrm{u})\varepsilon_{\mathrm{kl}}(\mathrm{u}) = \Lambda \tag{11.7}$$

which expresses that the strain energy density-like left-hand side term is constant and equal to Λ for all intermediate densities. This is thus a condition that is similar to the fully stressed design condition in plastic design. As we expect areas with high energy to be too low on stiffness we devise the following fix-point type update scheme for the density [3]:

$$\rho_{K+1} = \begin{cases} max\{(1-\varsigma)\rho_K, \rho_{min}\}, \text{ if } \rho_K B_K^{\eta} \leq max\{(1-\varsigma)\rho_K, \rho_{min}\} \\ min\{(1+\varsigma)\rho_K, 1\}, \text{ if } min\{(1+\varsigma)\rho_K, 1\} \leq \rho_K B_K^{\eta} \\ \rho_K B_K^{\eta} \text{ else} \end{cases} \tag{11.8}$$

Here ρ_K denotes the value of the density variable at iteration step K, and B_K is given by the expression

$$B_K = \Lambda_K^{-1} p\rho(x)^{p-1} E^0_{ijkl}\varepsilon_{ij}(u_K)\varepsilon_{kl}(u_K), \tag{11.9}$$

where u_K is the displacement field at at the iteration step K, determined from the equilibrium equation and dependent on ρ_K. Note that a (local) optimum is reached if $B_K = 1$. The update

scheme (11.8) adds material to areas with a specific strain energy that is higher than Λ (that is, when $B_K > 1$) and removes it if the energy is below this value; this only takes place if the update does not violate the bounds on p. One can see that Λ is proportional (by a factor p) to the average strain energy density of the part of the structure that is given by intermediate values of the density. The variable η in (11.8) is a tuning parameter and ς a move limit. Both η and ς controls the changes that can happen at each iteration step and they can be made adjustable for efficiency of the method. Note that the update ρ_{K+1} depends on the present value of the Lagrange multiplier Λ, and thus Λ should be adjusted in an inner iteration loop in order to satisfy the active volume constraint. It is readily seen that the volume of the updated values of the densities is a continuous and decreasing function of the multiplier Λ. Moreover, the volume is strictly decreasing in the interesting intervals, where the bounds on the densities are not active in all points (elements of a FEM discretization). This means that we can uniquely determine the value of Λ, using a bisection method or a Newton method. The values of η and ς are chosen by experiment, in order to obtain a suitable rapid and stable convergence of the iteration scheme. A typical useful value of η and ς is 0.5 and 0.2, respectively.

The type of algorithm described above has been used to great effect in a large number of structural topology design studies and is well established as an effective method for solving large scale problems. The effectiveness of the algorithm comes from the fact that each design variable is updated independently of the update of the other design variables, except for the rescaling that has to take place for satisfying the volume constraint. The algorithm can be generalized to quite a number of structural optimization settings, but it is not always straightforward. For cases where for example constraints of a non-structural nature should be considered (e.g., representing geometry considerations), when non-self-adjoint problems are considered or where physical intuition is limited, the use of a mathematical programming method can be a more direct way to obtain results. Typically, this will be computationally more costly, but a careful choice of algorithm can make this approach as efficient as the optimality criteria method.

This optimality criteria method is very effective and can be extended to another problem formulation; it needs few programming effort in case of having a finite element code. In those cases where the application of the optimality criteria is not possible, we can use mathematical programming method.
The prescribed volume V^* can have predominant effect as it can be seen on Figure 11.5 so the designer must be carefully decide by choosing this parameter.

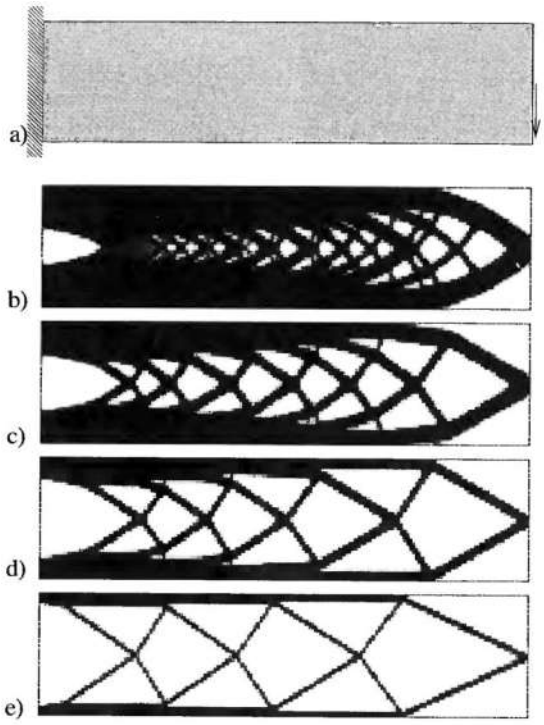

Figure 11.5. Effect of prescribed volume fraction b) 80%; c) 60%; d) 40%; e) 20%

Ensuring convergence with mesh refinement

The basic algorithm does not ensure a unique solution with mesh refinement (Figure 11.6). There is three main, principally different techniques to ensure the convergence in case of mesh refinement:

- control of perimeter or prescribing minimal member size
- reduction of parameter space (coarser mesh on design space)
- filtering methods (sensitivity or density filtering).

Having greater domain with intermediate density can be avoided effectively applying the simple *p* penalty parameter. It is very useful because of interpreting a solution with intermediate density is not easy in every case (in some case we can suppose a porous, or thinner or weaker material).

The method can result a **checker board** like pattern, this has also very few connection to real structure. For solving both problems we can apply projection methods and sensitivity filtering.

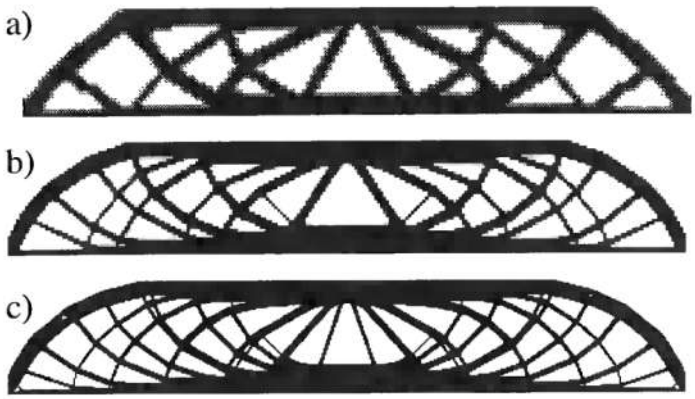

Figure 11.6. Effect of mesh refinement to the optimum a) 2700 b) 4800 c) 17200 elements

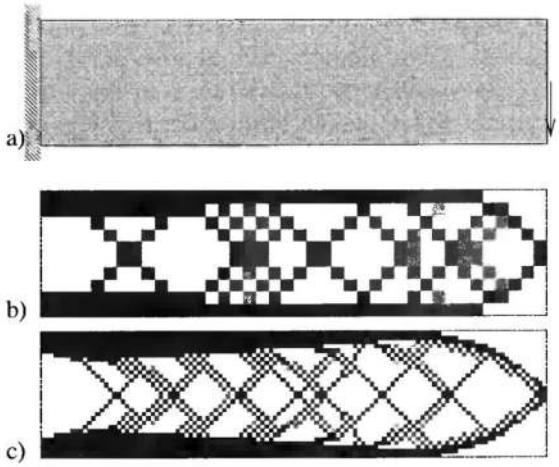

Figure 11.7. Checker board like solution for 400 and 6400 elements

Naturally, applying more than one load case changes the characteristic of the solution (Figure 11.8). Figure c) and d) shows the optimized topologies for all loads in one load case. In Figure e) and f) we can observe the optimized topologies for multiple loading cases. It is seen that single load problems result in instable structures based on square frames whereas multi load case problems results in stable structures based on triangular frames.

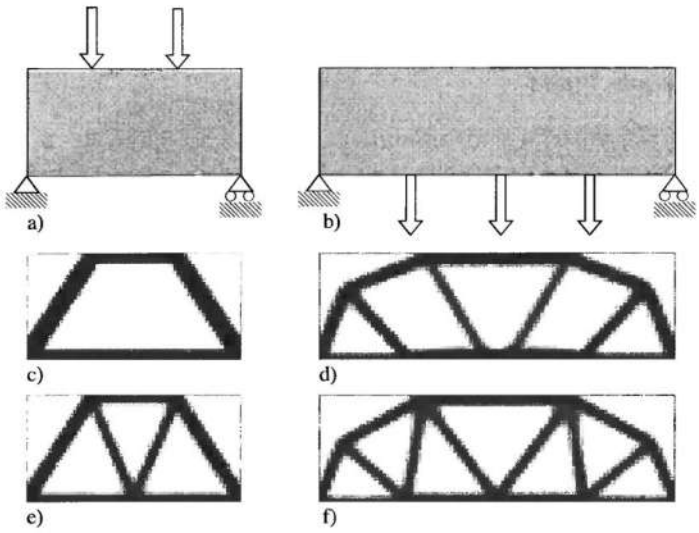

Figure 11.8. Optimum for one and for multiple load cases

The problem solving sequence with SIMP method is as here follows:

- define the initial design domain, the boundary value problem, with homogeneous material distribution,
- compute the displacements and strains for the actual design density with the finite element method,
- compute the compliance for the actual design,
- in case of having no significant changes or achieving the optimum criteria stop else
- compute the new material distribution due to equations (11.7) and (11.8) (and with an inner loop the value of Λ Lagrange multiplicator due to the volume constraint),
- repeat the procedure from the second step.

In the case applying mathematical programming method to compute the optimum, we need also the sensitivities of objective function and optimization constraints. Writing the problem in discrete form after (11.3) and (11.4), the sensitivities can be easily calculated with the adjoint method for the compliance problem as in this case the adjoint variable is identical with the displacement so

$$\frac{\partial c}{\partial \rho_e} = -p\rho_e^{p-1}\,\boldsymbol{u}^T\boldsymbol{K}_e\boldsymbol{u} \tag{11.10}$$

The sensitivity does not depend directly on the other elements, only implicit way through the displacements and it can be easily computed. As the sensitivity takes negative values, the intuition as adding material will decrease the compliance and increase the stiffness. Having a fine design, we get high number of design variables (one for each element) which can be handled with Method of Moving Asymptotes (MMA). The algorithm and programming effort is similar to the previously demonstrated optimum criteria method. The utmost time cosuming part of the calculation is the structural analysis.

11.6. Level set methods(LSM) [11.3]

The fast marching method and the level set method are numerical techniques which follows the change of an interface coupling it to a level set function. In level set-based structural op-

timization methods, complex shape and topological changes can be handled and the obtained optimal structures are free from greyscales, since the structural boundaries are represented as the iso-surface of the level set function. These relatively new structural optimization methods overcome the problems of checkerboard patterns and greyscales. The interface can have unsmooth shape with edges and corners, its topology can be divided or rejoined without having computational problems because the level set function can follow this changes on a smooth way.
Rather than follow the interface itself, the Fast Marching Method makes use of stationary approach to the problem. Let us lay a grid laid down on top of the problem. A time like function *T(x,y)* is ordered to the moving interface: at each grid point *T, T(x,y)* gives the time at which the front crosses the point *(x,y).*
As an example by Sethian et al., suppose the initial disturbance is a circle propagating outwards. The original region (the blue one on the left below) propagates outwards, crossing over each of the timing spots. The function *T(x,y)* gives a cone-shaped surface, which is shown on the right. This surface intersects the *xy* plane exactly where the curve is initially. At any height *T* the surface gives the set of points reached at time *T*. The surface on the right below is called the arrival time surface, because it gives the arrival time.

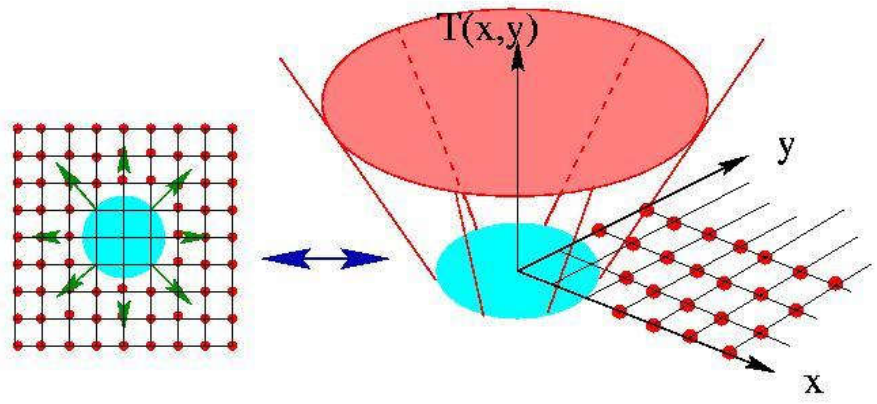

Figure 11.9. Interpretation of the fast marching method

Beneficially, the numerical calculations are made on a fix grid (Eulerian approximation).

The level set function is more general: the interface is ordered to a signed distance function and the initial shape is ordered to the zero level. The initial value problem for the moving of the level set function corresponds to the Hamilton-Jacobi equation.

In the following example a bending plate is given on Figure 11.10. The red arrow denotes the load, green squares the fixation. The design boundary is drawn with broken line. Stress distribution is shown on Figure 11.11.

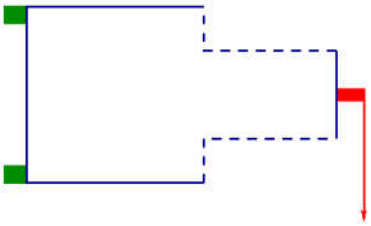

Figure 11.10. Problem definition

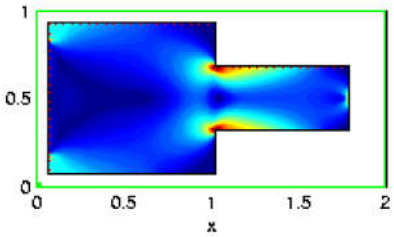

Figure 11.11. Stress distribution of start design

A hole is inserted on the low stressed zone and the stresses are recalculated; the hole is enlarged in the direction to the low stressed zone.

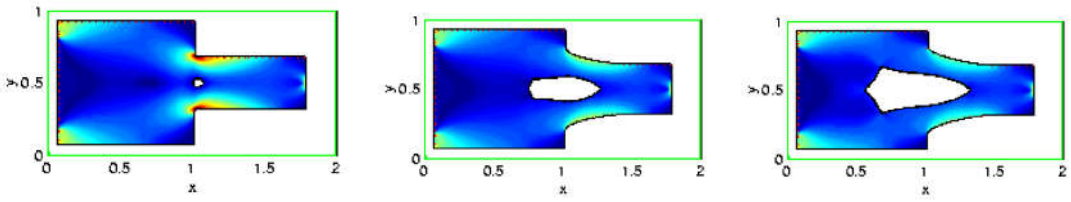

Figure 11.12. Operation of level set method

11.7. Evolutionary Structural Optimization (ESO) [11.7][11.8][11.9]

The evolutionary methods based on engineering intuition: they extract domains not participating in the load transfer and add material to strengthen the weak places (bidirectional methods). Initial finite element model for the **Hard-Kill (HK)** Method can be a structured mesh of a given part; the elimination of elements follows with the reduction to near zero of its material properties. Benefit of the method is there is no need for remeshing and every kind of problem can be solved even for more load cases. Its' drawback is the zigzagging mesh and if the step-width is chosen inappropriately, the supported or the loaded surface can be vanished. Need for mesh refinement is also a problematic task. Metamophic development is a bidirectional version of evolutionary methods, which adapts the amount of added/extracted material dynamically to the structural responses in every iteration steps, accelerating the convergence.

11.8. Nonprofit software tools to obtain the optimal topology

In the internet there are available some free software to demonstrate how to estimate the optimal layout of a design using topology optimization techniques. In this section the capability and problem solving sequence of TOPOPT and TOPOSTRUCT programs will be demonstrated. Both of them search the optimal material distribution (SIMP method) on fixed mesh of an initially given brick shaped domain.

Steps of compliance minimization:

- Choosing between 2D or 3D
- giving the size and resolution of design space
- Prescription of supports and loads
- Volume fraction up 0.1 to 0.2

- Maximal iteration limit and penalty parameter
- solving the optimization
- evaluation the results
- adapt and forward the results.

11.8.1. TOPOPT

TOPOPT is developed on the Technical University of Denmark, in a cooperation of Faculty of Mathematics and Mechanical Engineering for supporting the theoretical developments and solving practical problems. The preprocessing the optimization problem takes place in a Java applet, the calculation runs on a remote computer. Results appear as animated gif on the local machine.

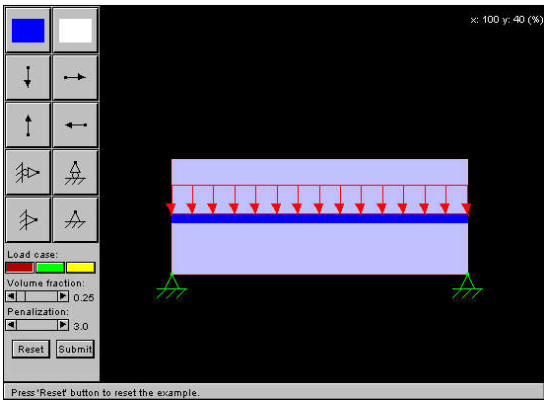

Figure 11.13. Definition of the optimization problem with TOPOPT

In the case program for solving 2 dimensional compliance problem design domain can be given with dragging the size of a light blue rectangle on the black background after its selection between 100x100 units (Figure 11.14.). Prescribing holes for occupancy of connected components or is possible thru selecting, adapting and dragging white rectangles while conservation of supported and/or loaded surfaces can be ensured using darkblue rectangles in arbitrary number.

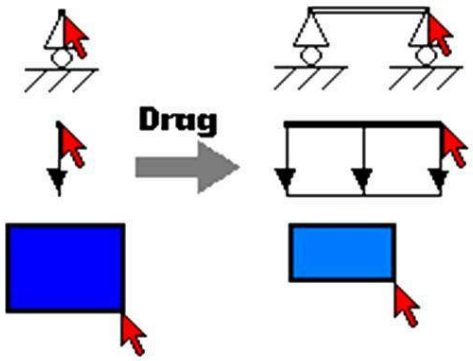

Figure 11.14. Prescribing tools for the boundary value problem

Symmetry should be utilized for achieving finer results. Mechanical problem can be also given by choosing the supports and loads in appropriate direction and applying them on the body thru dragging.

Multiply load case can also be applied up to 3 loadcases.

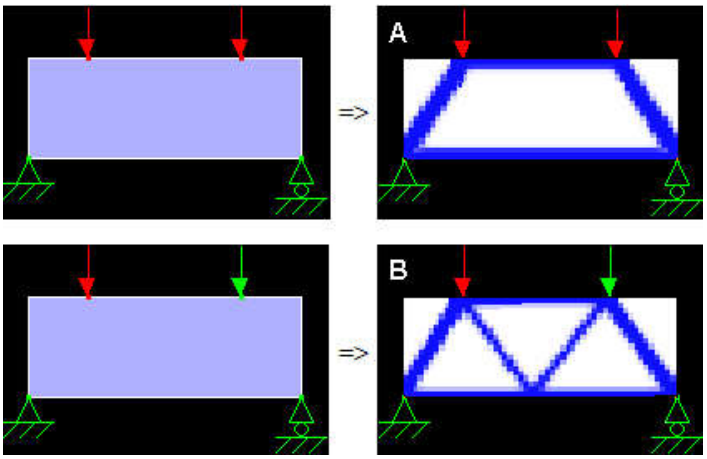

Figure 11.15. Topology optimization for more load cases 1.

On the top of Figure 11.15. the two loads act simultaneously, on the bottom apart from each other. The difference of the solution can be observed on the right hand side.

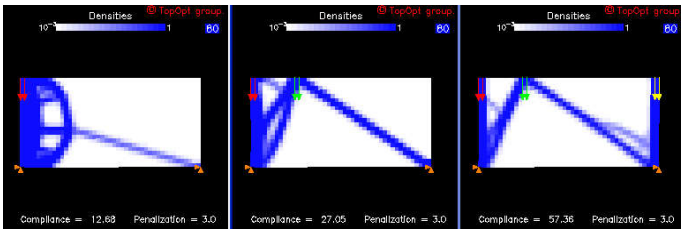

Figure 11.16. Topology optimization for more load cases 2.

Effect of additional load cases can be followed on Figure 11.16.
Volume fraction (up 0 to 1) and penalty parameter (between 1 and 3) can be adjusted with the slides. Limits of the 2 dimensional programs are

- 1000 design variables
- 3 load cases
- 100 iteration

Mechanism design can be made on a similar way with another program. Compliance design can be performed for 3D problem; in that case the work can be saved and results can be transferred to a CAD software in .stl format.

11.8.2. TOPOSTRUCT

Topostruct program was developed by Panagiotis Michalatos and Sawako Kaijima in 2008 for getting acquainted with topology optimization. The model can be given in 2 or 3 dimension, the design space is rectangular or brick.

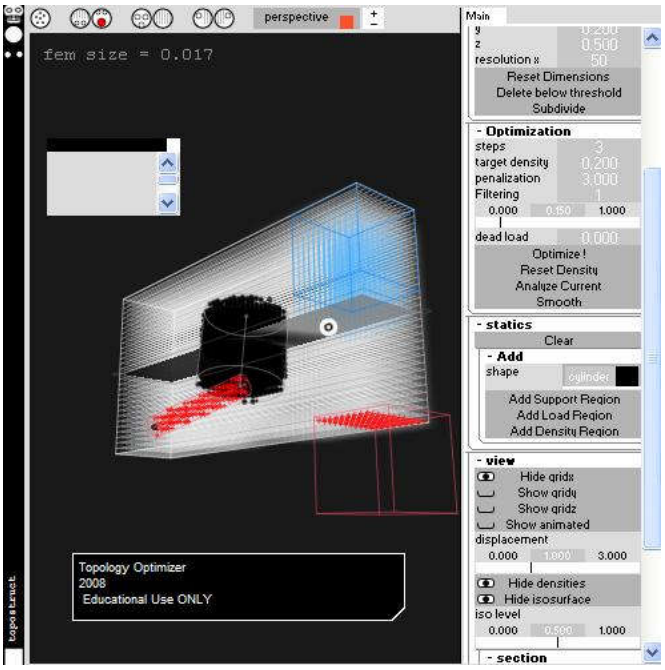

Figure 11.17. Topostruct program

Additional facility to TOPOPT, that the shape of the selected group of nodes for boundary value definition can be sphere or cylinder with axis in arbitrary direction. Model and results of structural analysis can be visually controlled and animated. The optimal result can be manipulated: it can be smoothed and elements can be eliminated under a given density level. Multiply loadcase can not be examined.
More detail can be found in the user manual. Operation can be understood in 2 dimensional és 3 dimensional examples.

11.9. Summary

Topology optimization searches the optimal layout for a global characteristic as compliance or natural frequency of a structural part, for a given material utilization and considering manufacturing constraints as extrusion, symmetry, draw direction by casting. The method should be able to find the optimal solution independent from the initial design guess.
An intuitive way for finding optimal layout is change the design due to engineering decision (cutting out low stressed part and strengthen the high stressed zones). These methods are called evolutionary methods. Such methods highly depend on the choice of parameters, so they are not stable and sometimes delivers unusable solution. The continuous topological changes should rather be followed with methods searching the optimal shape on an extended fix domain. The design variables able to follow these changes continuously could be

- geometrical parameters bonded to microstructure as in the homogenization method
- material like parameter as in the SIMP method
- bonded to a time like parameter after Euler formula, so the moving of the surface will be given in the initial coordinates as in the level set method and in the phase field method.

Numerical problem could arise and should be avoided as

- numerical instability
- dependency from mesh
- dependency from initial design
- checkerboard like solution
- greyscales or porous domains without any practical interpretation
- solution with manufacturing difficulties or with high manufacturing costs.

Solution of the above mentioned problem can be also multitude:

- geometrical limit as perimeter control or minimal member size control
- filtering techniques as sensitivity filtering
- using different mesh for design and analysis
- using manufacturing constraints as draw direction, extrusion direction, circular or axial symmetry
- further stabilizing conditions as giving a fictive limit surface energy or entropy like condition

Specialists are continuously searching the most suitable answers to all of these questions and extending the method for solving further type of problems. Nevertheless, results of a topology optimization should not interpreted directly as a design for manufacturing, interpretation leaves a task of design engineer, especially having a complex design problem. The optimal solution got for global condition should be further refined by an additional shape optimization step for local conditions as stress.

11.10. Literature

[11.1] Rozvany GIN. Exact analytical solutions for some popular benchmark problems in topology optimization. Structural Optimization 1998;15: 42-48

[11.2] Bendsøe MP, Kikuchi, N. Generating optimal topologies in structural design using a homogenization method. Computer Methods in Applied Mechanics and Engineering 1988;71:197-224.

[11.3] Bendsoe MP. Optimal shape design as a material distribution problem. Structural and Multidisciplinary Optimization. 1989;1:193–202.

[11.4] Sethian JA, Wiegmann A. Structural boundary design via level set and immersed interface methods. J Comput Phys 2000;163(2):489–528.

[11.5] Norato J, Bendsøe MP, Haber RB, Tortorelli, DA. Topological derivative method for topology optimization. Structural and Multidisciplinary Optimization 2007;33: 375-386.

[11.6] Takezawa A, Nishiwaki S, Kitamura M: Shape and topology optimization based on the phase field method and sensitivity analysis Journal of Computational Physics 229 (2010) 2697–2718

[11.7] Xie YM, Steven GP. A simple evolutionary procedure for structural optimization, Computers and Structures 1993;49: 885–896.

[11.8] Querin OM, Steven GP, Xie YM. Evolutionary structural optimization (ESO) using a bidirectional algorithm. Eng Comput 1998;15:1031–1048.

[11.9] Sauter J. CAOS oder die Suche nach der optimalen Bauteilform durch eine effiziente Gestaltoptimierungsstrategie. FEM '91, IKOSS CONGRESS, Tagungsband (S. 159-187), Baden-Baden, November 1991

[11.10] Reynolds D, McConnachie J, Bettess P, Christie WC, Bull JW. Reverse adaptivity—a new evolutionary tool for structural optimization. Int. J Numer Methods Engrg 1999;45 :529–552.

[11.11] Liu JS, Parks JT, Clarkson PJ: Optimization of Turbine Disk Profiles by Metamorphic Development. ASME Journal of Mechanical Design 2002;192-200

[11.12] Eschenauer HA, Kobelev HA, Schumacher A. Bubble method for topology and shape optimization of structures. Struct Optim 1994;8:142–151

[11.13] Victoria M, Marti P,Querin OM. Topology design of two-dimensional continuum structures using isolines. Computers & Structures 2009;87(1-2):101-109.

[11.14] Rozvany GIN. A critical review of established methods of structural topology optimization. Struct Multidisc Optim 2009;37:217–237.

[11.15] Bendsøe MP, Sigmund O. Topology Optimization: Theory, Methods, and Applications. Berlin, Heidelberg: Springer 2003

11.11. Questions

1. Which topology optimization methods can be used for more type of problems or for multidisciplinary problems?
2. What are benefits of the mathematical algorithms?
3. What designer decisions should be made before topology optimization?
4. What designer decisions should be made after topology optimization?