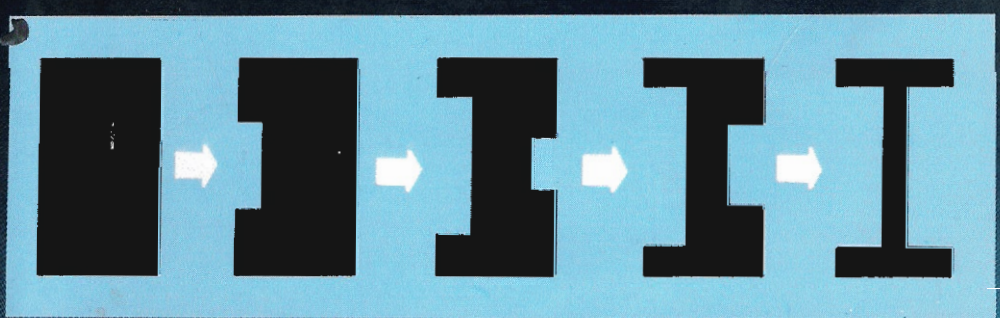
Eastern Economy Edition

OPTIMIZATION FOR ENGINEERING DESIGN

Algorithms and Examples



Kalyanmoy Deb

PHI

Optimization for Engineering Design

Algorithms and Examples

KALYANMOY DEB

Department of Mechanical Engineering
Indian Institute of Technology Kanpur

Prentice-Hall of India Private Limited

New Delhi - 110 001

2005

Rs. 195.00

OPTIMIZATION FOR ENGINEERING DESIGN: Algorithms and Examples
by Kalyanmoy Deb

ISBN-81-203-0943-X

Eighth Printing ... ... **July, 2005**

Published by Asoke K. Ghosh, Prentice-Hall of India Private Limited, M-97, Connaught Circus, New Delhi-110001 and Printed by Rajkamal Electric Press, B-35/9, G.T. Karnal Road Industrial Area, Delhi-110033.

To

MY PARENTS

Contents

Preface

Many engineers and researchers in industries and academics face difficulty in understanding the role of optimization in engineering design. To many of them, optimization is an esoteric technique used only in mathematics and operations research related activities. With the advent of computers, optimization has become a part of computer-aided design activities. It is primarily being used in those design activities in which the goal is not only to achieve just a feasible design, but also a design objective. In most engineering design activities, the design objective could be simply to minimize the cost of production or to maximize the efficiency of production. An optimization algorithm is a procedure which is executed iteratively by comparing various solutions till the optimum or a satisfactory solution is found. In many industrial design activities, optimization is achieved indirectly by comparing a few chosen design solutions and accepting the best solution. This simplistic approach never guarantees an optimal solution. On the contrary, optimization algorithms begin with one or more design solutions supplied by the user and then iteratively check new design solutions in order to achieve the true optimum solution. In this book, I have put together and discussed a few popular optimization algorithms and demonstrated their working principles by hand-simulating on a simple example problem. Some working computer codes are also appended for limited use.

There are two distinct types of optimization algorithms which are in use today. First, there are algorithms which are deterministic, with specific rules for moving from one solution to the other. These algorithms have been in use for quite some time and have been successfully applied to many engineering design problems. Secondly, there are algorithms which are stochastic in nature, with probabilistic transition rules. These algorithms are comparatively new and are gaining popularity due to certain properties which the deterministic

algorithms do not have. In this book, probably for the first time, an attempt has been made to present both these types of algorithms in a single volume. Because of the growing complexity in engineering design problems, the designer can no longer afford to rely on a particular method. The designer must know the advantages and limitations of various methods and choose the one that is more efficient to the problem at hand.

An important aspect of the optimal design process is the formulation of the design problem in a mathematical format which is acceptable to an optimization algorithm. However, there is no unique way of formulating every engineering design problem. To illustrate the variations encountered in the formulation process, I have presented four different engineering design problems in Chapter 1. Optimization problems usually contain multiple design variables, but I have begun by first presenting a number of single-variable function optimization algorithms in Chapter 2. The working principles of these algorithms are simpler and, therefore, easier to understand. Besides, these algorithms are used in multivariable optimization algorithms as unidirectional search methods. Chapter 3 presents a number of algorithms for optimizing unconstrained objective functions having multiple variables. Chapter 4 is an important one in that it discusses a number of algorithms for solving constrained optimization problems—most engineering design optimization problems are constrained. Chapter 5 deals with two specialized algorithms for solving integer programming problems and geometric programming problems. Two nontraditional optimization algorithms, which are very different in principle than the above algorithms, are covered in Chapter 6. Genetic algorithms—search and optimization algorithms that mimic natural evolution and genetics—are potential optimization algorithms and have been applied to many engineering design problems in the recent past. Due to their population approach and parallel processing, these algorithms have been able to obtain global optimal solutions in complex optimization problems. Simulated annealing method mimics the cooling phenomenon of molten metals. Due to its inherent stochastic approach and availability of a convergence proof, this technique has also been used in many engineering design problems. Chapter 6 also discusses the issue of finding the global optimal solution in a multioptimal problem, where the problem contains a number of local and global optimal solutions and the objective is to find the global optimal solution. To compare the power of various algorithms, one of the traditional constrained optimization techniques is compared with both the nontraditional optimization techniques in a multioptimal problem.

Some algorithms in Chapter 4 use linear programming methods, which are usually taught in operations research and transportation engineering related courses. Sometimes, linear programming methods are also taught in first or second-year undergraduate engineering courses. Thus, a detailed discussion of linear programming methods is avoided in this book. Instead, a brief analysis of the simplex search technique of the linear programming method is given in Appendix A.

The algorithms are presented in a step-by-step format so that they can be easily understood and coded in a computer language. The working principle of each algorithm is also illustrated by showing hand calculations up to a few iterations of the algorithms on a numerical optimization problem. The hand calculations provide a better insight into the working of the optimization algorithms. Moreover, in order to compare the efficiency of different algorithms, as far as possible, the same numerical example is chosen for each algorithm.

Most of the chapters contain at least one working computer code, implementing optimization algorithms presented in the chapter. The computer codes are written in FORTRAN programming language and sample simulation runs obtained under the Microsoft FORTRAN compiler on a PC platform are presented. These codes are also tested with a Unix FORTRAN compiler on a SUN machine. They demonstrate the ease and simplicity with which other optimization algorithms can also be coded. The computer codes presented in the text can be available by sending an e-mail to the author at `deb@iitk.ernet.in`.

The primary objective of this book is to introduce different optimization algorithms to students and design engineers, and provide them with a few computer codes for easy understanding. The mathematical treatment of the algorithms is kept at a less rigorous level so that the text can be used as an introductory book on optimization by design engineers as well as practitioners in industries and by the undergraduate and postgraduate students of engineering institutions. An elementary knowledge of matrix algebra and calculus would be sufficient for understanding most of the materials presented in the book.

Instructors may find this text useful in explaining optimization algorithms and solving numerical examples in the class, although occasional reference to a more theoretical treatment on optimization may be helpful. The best way to utilize this book is to begin with Chapter 1. This chapter helps the reader to correlate the design problems to the optimization problems already discussed.

Thereafter, subsequent chapters may be read one at a time. To have a better understanding of the algorithms, the reader must follow the steps of the solved exercise problems as they are outlined in the given algorithm. Then, the progress of each algorithm may be understood by referring to the accompanying figure. For better comprehension, the reader may use the FORTRAN code given at the end of the chapters to solve the example problems.

Any comments and suggestions for improving the text would be always welcome.

Kalyanmoy Deb

Acknowledgements

The person who introduced me to the field of optimization and who has had a significant role in moulding my career is Professor David E. Goldberg of the University of Illinois at Urbana-Champaign. On a lunch table, he once made me understand that probably the most effective way of communicating one's ideas is through books. That discussion certainly motivated me in taking up this project. The main inspiration for writing this book came from Professor Amitabha Ghosh, Mechanical Engineering Department, IIT Kanpur, when in one tutorial class I showed him the fifty-page handout I prepared for my postgraduate course entitled "Optimization Methods in Engineering Design". Professor Ghosh looked at the handout and encouraged me to revise it in the form of a textbook. Although it took me about an year-and-half to execute that revision, I have enjoyed every bit of my experience.

Most of the algorithms presented in this text are collected from various books and research papers related to engineering design optimization. My sincere thanks and appreciation are due to all authors of those books and papers. I have been particularly influenced by the concise and algorithmic approach adopted in the book entitled 'Engineering Optimization–Methods and Applications' by G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. Many algorithms presented here are modified abstractions from that book.

I am also grateful to Professor Brahma Deo and Dr. Partha Chakroborty for their valuable comments which significantly improved the contents of this book. The computer facility of the Computer Aided Design (CAD) Project, generously provided by Professor Sanjay Dhande, is highly appreciable. My special thanks are due to two of my students N. Srinivas and Ram Bhusan Agrawal for helping me in drawing some of the diagrams and checking some exercise problems. The computer expertise provided by P. V. M. Rao, Samir Kulkarni, and Sailesh Srivastava in preparing one of

the computer codes is also appreciated. Discussions with Professors David Blank and M. P. Kapoor on different issues of optimization were also helpful. I am thankful to my colleagues and staff of the CAD Project for their constant support. It would have taken at least twice the time to complete this book, if I did not have the privilege to meet Dr. Subhransu Roy who generously provided me with his text-writing and graph plotting softwares. My visits to TELCO, TISCO, Hindustan Motors and Engineers India Ltd, and the discussions I had with many design engineers were valuable in writing some of the chapters. The financial assistance provided by the Continuing Education Centre at the Indian Institute of Technology Kanpur to partially compensate for the preparation of the manuscript is gratefully acknowledged. I also wish to thank the Publishers, Prentice-Hall of India, for the meticulous care they took in processing the book.

This book could not have been complete without the loving support and encouragement of my wife, Debjani. Her help in typing a significant portion of the manuscript, in proof-reading, and in preparing the diagrams has always kept me on schedule. Finally, I take this opportunity to express my gratitude to my parents and my loving affection to my brothers.

Indian Institute of Technology
Kanpur

Kalyanmoy Deb

1

Introduction

Optimization algorithms are becoming increasingly popular in engineering design activities, primarily because of the availability and affordability of high speed computers. They are extensively used in those engineering design problems where the emphasis is on maximizing or minimizing a certain goal. For example, optimization algorithms are routinely used in aerospace design activities to minimize the overall weight, simply because every element or component adds to the overall weight of the aircraft. Thus, the minimization of the weight of aircraft components is of major concern to aerospace designers. Chemical engineers, on the other hand, are interested in designing and operating a process plant for an optimum rate of production. Mechanical engineers design mechanical components for the purpose of achieving either a minimum manufacturing cost or a maximum component life. Production engineers are interested in designing optimum schedules of various machining operations to minimize the idle time of machines and the overall job completion time. Civil engineers are involved in designing buildings, bridges, dams, and other structures in order to achieve a minimum overall cost or maximum safety or both. Electrical engineers are interested in designing communication networks so as to achieve minimum time for communication from one node to another.

All the above-mentioned tasks involve either minimization or maximization (collectively known as optimization) of an objective. It is clear from the spectrum of the above problems that it is difficult to discuss the formulation of various engineering optimization problems in a single book. Fortunately, a designer specialized in a particular

design is usually more informed about different factors governing that design than anyone else. Thus, as far as the formulation of the optimal problem is concerned, the designer can acquire it with some practice. However, every designer should know a few aspects of the formulation procedure which would help him or her to choose a proper optimization algorithm for the chosen optimal design problem. This requires a knowledge about the working principles of different optimization methods. In subsequent chapters, we discuss various optimization methods which would hopefully provide some of that knowledge. In this chapter, we demonstrate the optimal problem formulation procedures of four different engineering optimal design problems.

1.1 Optimal Problem Formulation

In many industrial design activities, a naive optimal design is achieved by comparing a few (limited up to ten or so) alternative design solutions created by using a priori problem knowledge. In such an activity, the feasibility of each design solution is first investigated. Thereafter, an estimate of the underlying objective (cost, profit, etc.) of each design solution is computed and the best design solution is adopted. This naive method is often followed because of the time and resource limitations. But, in many cases this method is followed simply because of the lack of knowledge of the existing optimization procedures. But whatever may be the reason, the purpose of either achieving a quality product or of achieving a competing product is never guaranteed to be fulfilled with the above naive method. Optimization algorithms described in this book provide systematic and efficient ways of creating and comparing new design solutions in order to achieve an optimal design. Since an optimization algorithm requires comparison of a number of design solutions, it is usually time-consuming and computationally expensive. Thus, the optimization procedure must only be used in those problems where there is a definite need of achieving a quality product or a competitive product. It is expected that the design solution obtained through an optimization procedure is better than other solutions in terms of the chosen objective—cost, efficiency, safety, or others.

We begin our discussion with the formulation procedure by mentioning that it is almost impossible to apply a single formulation procedure for all engineering design problems. Since the objective in a design problem and the associated design parameters vary from product to product, different techniques need to be used in different problems. The purpose of the formulation procedure is to create a

mathematical model of the optimal design problem, which then can be solved using an optimization algorithm. Since an optimization algorithm accepts an optimization problem in a particular format, every optimal design problem must be formulated in that format. In this section, we discuss different components of that format.

Figure 1.1 shows an outline of the steps usually involved in an optimal design formulation process. As mentioned earlier, the first

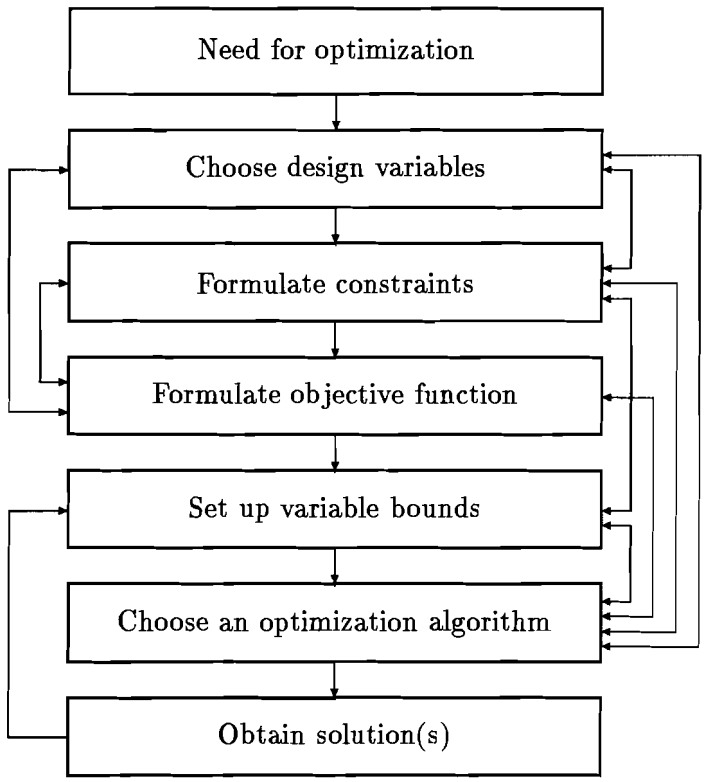


Figure 1.1 A flowchart of the optimal design procedure.

step is to realize the need for using optimization in a specific design problem. Thereafter, the designer needs to choose the important design variables associated with the design problem. The formulation of optimal design problems involves other considerations, such as constraints, objective function, and variable bounds. As shown in the figure, there is usually a hierarchy in the optimal design process; although one consideration may get influenced by the other. We discuss all these aspects in the following subsections.

1.1.1 Design variables

The formulation of an optimization problem begins with identifying the underlying design variables, which are primarily varied during the optimization process. A design problem usually involves many design parameters, of which some are highly sensitive to the proper working of the design. These parameters are called design variables in the parlance of optimization procedures. Other (not so important) design parameters usually remain fixed or vary in relation to the design variables. There is no rigid guideline to choose a priori the parameters which may be important in a problem, because one parameter may be more important with respect to minimizing the overall cost of the design, while it may be insignificant with respect to maximizing the life of the product. Thus, the choice of the important parameters in an optimization problem largely depends on the user. However, it is important to understand that the efficiency and speed of optimization algorithms depend, to a large extent, on the number of chosen design variables. In subsequent chapters, we shall discuss certain algorithms which work very efficiently when the number of design variables is small, but do not work that well for a large number of design variables. Thus, by selectively choosing the design variables, the efficacy of the optimization process can be increased. The first *thumb* rule of the formulation of an optimization problem is to choose as few design variables as possible. The outcome of that optimization procedure may indicate whether to include more design variables in a revised formulation or to replace some previously considered design variables with new design variables.

1.1.2 Constraints

Having chosen the design variables, the next task is to identify the constraints associated with the optimization problem. The constraints represent some functional relationships among the design variables and other design parameters satisfying certain physical phenomenon and certain resource limitations. Some of these considerations require that the design remain in static or dynamic equilibrium. In many mechanical and civil engineering problems, the constraints are formulated to satisfy stress and deflection limitations. Often, a component needs to be designed in such a way that it can be placed inside a fixed housing, thereby restricting the size of the component. There is, however, no unique way to formulate a constraint in all problems. The nature and number of constraints to be included in the formulation depend on the user. In many algorithms discussed in this book, it is not necessary to have an

explicit mathematical expression of a constraint; but an algorithm or a mechanism to calculate the constraint is mandatory. For example, a mechanical engineering component design problem may involve a constraint to restrain the maximum stress developed anywhere in the component to the strength of the material. In an irregular-shaped component, there may not exist an exact mathematical expression for the maximum stress developed in the component. A finite element simulation software may be necessary to compute the maximum stress. But the simulation procedure and the necessary input to the simulator and the output from the simulator must be understood at this step.

There are usually two types of constraints that emerge from most considerations. Either the constraints are of an inequality type or of an equality type. Inequality constraints state that the functional relationships among design variables are either greater than, smaller than, or equal to, a resource value. For example, the stress ($\sigma(x)$) developed anywhere in a component must be smaller than or equal to the allowable strength ($S_{\text{allowable}}$) of the material. Mathematically,

$$\sigma(x) \leq S_{\text{allowable}}.$$

Most of the constraints encountered in engineering design problems are of this type. Some constraints may be of greater-than-equal-to type: for example, the natural frequency ($\nu(x)$) of a system may required to be be greater than 2 Hz, or mathematically, $\nu(x) \geq 2$. Fortunately, one type of inequality constraints can be transformed into the other type by multiplying both sides by -1 or by interchanging the left and right sides. For example, the former constraint can be transformed into a greater-than-equal-to type by either $-\sigma(x) \geq -S_{\text{allowable}}$ or $S_{\text{allowable}} \geq \sigma(x)$.

Equality constraints state that the functional relationships should exactly match a resource value. For example, a constraint may require that the deflection ($\delta(x)$) of a point in the component must be exactly equal to 5 mm, or mathematically,

$$\delta(x) = 5.$$

Equality constraints are usually more difficult to handle and, therefore, need to be avoided wherever possible. If the functional relationships of equality constraints are simpler, it may be possible to reduce the number of design variables by using the equality constraints. In such a case, the equality constraints reduce the complexity of the problem, thereby making it easier for the optimization algorithms to solve the problem. In Chapter 4, we

discuss a number of algorithms which are specially designed to handle equality constraints. Fortunately, in many engineering design optimization problems, it may be possible to relax an equality constraint by including two inequality constraints. The above deflection equality constraint can be replaced by two constraints:

$$\delta(x) \geq 4,$$

$$\delta(x) \leq 6.$$

The exact deflection requirement of 5 mm is relaxed by allowing it to deflect anywhere between 4 mm to 6 mm. Although this formulation is inexact as far as the original requirement is concerned, this flexibility allows a smoother operation of the optimization algorithms. Thus, the second *thumb* rule in the formulation of optimization problems is that the number of complex equality constraints should be kept as low as possible.

1.1.3 Objective function

The third task in the formulation procedure is to find the objective function in terms of the design variables and other problem parameters. The common engineering objectives involve minimization of overall cost of manufacturing, or minimization of overall weight of a component, or maximization of net profit earned, or maximization total life of a product, or others. Although most of the above objectives can be quantified (expressed in a mathematical form), there are some objectives that may not be quantified easily. For example, the aesthetic aspect of a design, ride characteristics of a car suspension design, and reliability of a design are important objectives that one may be interested in maximizing in a design, but the exact mathematical formulation may not be possible. In such a case, usually an approximating mathematical expression is used. Moreover, in any real-world optimization problem, there could be more than one objective that the designer may want to optimize simultaneously. Even though a few multiobjective optimization algorithms exist in the literature (Chankong and Haimes, 1983), they are complex and computationally expensive. Thus, in most optimal design problem, multiple objectives are avoided. Instead, the designer chooses the most important objective as the objective function of the optimization problem, and the other objectives are included as constraints by restricting their values within a certain range. For example, consider an optimal truss structure

design problem. The designer may be interested in minimizing the overall weight of the structure and simultaneously be concerned in minimizing the deflection of a specific point in the truss. In the optimal problem formulation, the designer may like to use the weight of the truss (as a function of the cross-sections of the members) as the objective function and have a constraint with the deflection of the concerned point to be less than a specified limit. In general, the objective function is not required to be expressed in a mathematical form. A simulation package may be required to evaluate the objective function. But whatever may be the way to evaluate the objective function, it must be clearly understood.

The objective function can be of two types. Either the objective function is to be maximized or it has to be minimized. Unfortunately, the optimization algorithms are usually written either for minimization problems or for maximization problems. Although in some algorithms, some minor structural changes would enable to perform either minimization or maximization, this requires extensive knowledge of the algorithm. Moreover, if an optimization software is used for the simulation, the modified software needs to be compiled before it can be used for the simulation. Fortunately, the *duality* principle helps by allowing the same algorithm to be used for minimization or maximization with a minor change in the objective function instead of a change in the entire algorithm. If the algorithm is developed for solving a minimization problem, it can also be used to solve a maximization problem by simply multiplying the objective function by -1 and vice versa. For example, consider the maximization of the single-variable function $f(x) = x^2(1-x)$ shown by a solid line in Figure 1.2. The maximum point happens to be at $x^* = 0.667$. The duality principle suggests that the above problem is equivalent to minimizing the function $F(x) = -x^2(1-x)$, which is shown by a dashed line in Figure 1.2. The figure shows that the minimum point of the function $F(x)$ is also at $x^* = 0.667$. Thus, the optimum solution remains the same. But once we obtain the optimum solution by minimizing the function $F(x)$, we need to calculate the optimal function value of the original function $f(x)$ by multiplying $F(x)$ by -1.

1.1.4 Variable bounds

The final task of the formulation procedure is to set the minimum and the maximum bounds on each design variable. Certain optimization algorithms do not require this information. In these problems, the constraints completely surround the feasible region. Other problems

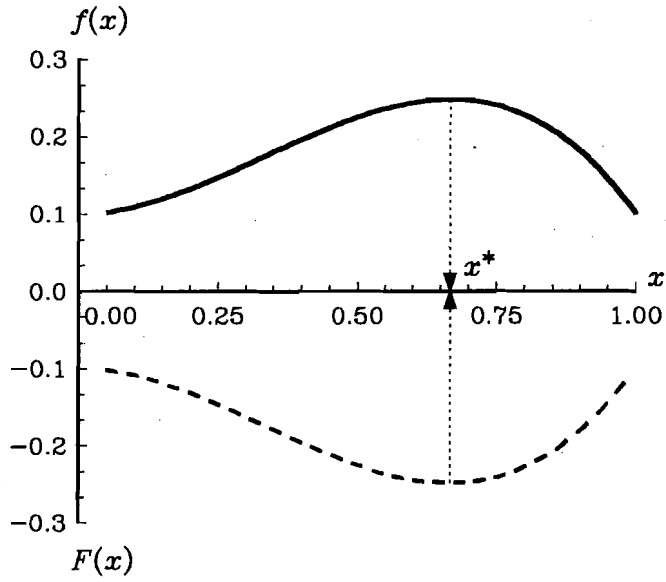


Figure 1.2 Illustration of the duality principle. The maximum point of $f(x)$ is the same as the minimum point of $F(x)$.

require this information in order to confine the search algorithm within these bounds. In general, all N design variables are restricted to lie within the minimum and the maximum bounds as follows:

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad \text{for } i = 1, 2, \ldots, N.$$

In any given problem, the determination of the variables bounds $x_i^{(L)}$ and $x_i^{(U)}$ may be difficult. One way to remedy this situation is to make a guess about the optimal solution and set the minimum and maximum bounds so that the optimal solution lies within these two bounds. After simulating the optimization algorithm once, if the optimal solution is found to lie within the chosen variable bounds, there is no problem. On the other hand, if any design variable corresponding to the optimal solution is found to lie on or near the minimum or the maximum bound, the chosen bound may not be correct. The chosen bound may be readjusted and the optimization algorithm may be simulated again. Although this strategy may seem to work only with linear problems, it has been found useful in many real-world engineering optimization problems.

After the above four tasks are completed, the optimization problem can be mathematically written in a special format, known as *nonlinear programming* (NLP) format. Denoting the design variables

as a column vector[1] $x = (x_1, x_2, \ldots, x_N)^T$, the objective function as a scalar quantity $f(x)$, J inequality constraints as $g_j(x) \geq 0$, and K equality constraints as $h_k(x) = 0$, we write the NLP problem:

$$\left.\begin{aligned} &\text{Minimize} \quad f(x) \\ &\text{subject to} \\ &g_j(x) \geq 0, \qquad j = 1, 2, \ldots, J; \\ &h_k(x) = 0, \qquad k = 1, 2, \ldots, K; \\ &x_i^{(L)} \leq x_i \leq x_i^{(U)}, \qquad i = 1, 2, \ldots, N. \end{aligned}\right\} \tag{1.1}$$

Note that the above formulation can represent a formulation for maximization problems by using the duality principle and can represent a formulation for problems with the lesser-than-equal-to type inequality constraints by using the techniques described earlier. However, the optimization algorithm used to solve the above NLP problem depends on the type of the objective function and constraints. It is important to note that the constraints must be written in a way so that the right-side of the inequality or equality sign is zero.

It is worth mentioning here that all the above four tasks are not independent of each other. While formulating the objective function, the designer may decide to include or delete some constraints. In many problems, while the constraints are being formulated, it is necessary to add some artificial design variables, which make the overall formulation easier. The update of the design variables, the constraints, the objective function, and the variable bounds may continue for a few iterations until the designer is finally satisfied with a reasonable formulation. Certain possible iterations are shown in Figure 1.1. We may mention here that this update also depends on the knowledge of the optimization algorithms to be used to solve the problem. But this requires some practice of using optimization algorithms before such input may be incorporated into the formulation procedure. Nevertheless, after the optimization problem is formulated, an optimization algorithm is chosen and an optimal solution of the NLP problem is obtained. We now illustrate the above four steps of the formulation procedure in four engineering optimal design problems.

[1]The representation of the design variables in the above column vector helps to achieve some matrix operations in certain multivariable optimization methods described in Chapters 3 and 4.

1.2 Engineering Optimization Problems

Because of the variety of engineering design problems, it is not possible to discuss the formulation of various optimization problems that are usually encountered in engineering design. To illustrate the formulation procedure, we formulate four different optimization problems. The first is a truss structure design problem, the second is a chemical reactor design problem, the third problem is an interesting transport scheduling problem, and the fourth problem involves the optimal design of a car suspension system. The formulation procedures show different considerations often adopted in formulating engineering optimization problems.

1.2.1 Optimal design of a truss structure

A truss structure is used in many civil engineering applications including bridges, buildings, and roofs. There are two different types of optimization problems in a truss structure design. Firstly, the topology of the truss structure (the connectivity of the elements in a truss) could be an optimization problem. In this problem, the objective is to find the optimal connectivity of truss elements so as to achieve the minimal cost of materials and construction. Secondly, once the optimal layout of the truss is known, the determination of every element cross-section is another optimization problem. In this problem, the objective is to find the optimal cross-sections of all elements in order to achieve a minimum cost of materials and construction. Although both these problems attempt to achieve the same objective, the search space and the optimization algorithm required to solve each problem are different. Here, we discuss the latter problem only. However, there exist certain algorithms which can be used to solve both the above problems simultaneously. We discuss more about these algorithms in Chapter 6.

Consider the seven-bar truss structure shown in Figure 1.3. The loading is also shown in the figure. The length of the members $AC = CE = \ell = 1$ m. Once the connectivity of the truss is given, the cross-sectional area and the material properties of the members are the design parameters. Let us choose the cross-sectional area of members as the design variables for this problem. There are seven design variables, each specifying the cross-section of a member (A_1 to A_7). Using the symmetry of the truss structure and loading, we observe that for the optimal solution, $A_7 = A_1$, $A_6 = A_2$, and $A_5 = A_3$. Thus, there are practically four design variables (A_1 to A_4). This completes the first task of the optimization

procedure.

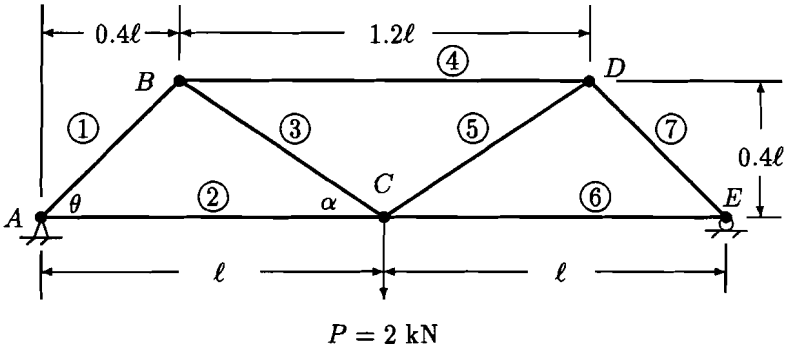


Figure 1.3 A typical seven-bar truss structure.

The next task is to formulate the constraints. In order for the truss to carry the given load $P = 2$ kN, the tensile and compressive stress generated in each member must not be more than the corresponding allowable strength S_{yt} and S_{yc} of the material. Let us assume that the material strength for all elements is $S_{yt} = S_{yc} = 500$ MPa and the modulus of elasticity $E = 200$ GPa. For the given load, we can compute the axial force generated in each element (Table 1.1). The positive force signifies tensile load and the negative force signifies compressive load acting on the member. Thereafter,

Table 1.1 Axial Force in Each Member of the Truss

Member	Force	Member	Force
AB	$-\frac{P}{2}\csc\theta$	BC	$+\frac{P}{2}\csc\alpha$
AC	$+\frac{P}{2}\cot\theta$	BD	$-\frac{P}{2}(\cot\theta+\cot\alpha)$

the axial stress can be calculated by dividing the axial load by the cross-sectional area of that member. Thus, the first set of constraints can be written as

$$\frac{P\csc\theta}{2A_1} \le S_{yc},$$

$$\frac{P\cot\theta}{2A_2} \le S_{yt},$$

$$\frac{P \csc \alpha}{2A_3} \leq S_{yt},$$

$$\frac{P}{2A_4}(\cot \theta + \cot \alpha) \leq S_{yc}.$$

In the above truss structure, $\tan \theta = 1.0$ and $\tan \alpha = 2/3$. The other set of constraints arises from the stability consideration of the compression members AB, BD, and DE. Realizing that each of these members is connected by pin joints, we can write the Euler buckling conditions for the axial load in members AB and BD (Shigley, 1986):

$$\frac{P}{2 \sin \theta} \leq \frac{\pi E A_1^2}{1.281 \ell^2},$$

$$\frac{P}{2}(\cot \theta + \cot \alpha) \leq \frac{\pi E A_4^2}{5.76 \ell^2}.$$

In most structures, deflection is a major consideration. In the above truss structure, let us assume that the maximum vertical deflection at C is $\delta_{\max} = 2$ mm. By using Castigliano's theorem (Timoshenko, 1986), we obtain the deflection constraint:

$$\frac{P\ell}{E}\left(\frac{0.566}{A_1} + \frac{0.500}{A_2} + \frac{2.236}{A_3} + \frac{2.700}{A_4}\right) \leq \delta_{\max}.$$

All the above constraints are of less-than-equal-to type. Once the constraints are formulated, the next task is to formulate the objective function. In this problem, we are interested in minimizing the weight of the truss structure. Since we have assumed the same material for all members, the minimization of the total volume of material will yield the same optimal solution as the minimization of the total weight. Thus, we write the objective function as

$$\text{Minimize} \quad 1.132A_1\ell + 2A_2\ell + 1.789A_3\ell + 1.2A_4\ell.$$

The fourth task is to set some lower and upper bounds for the four cross-sectional areas. We may choose to make all four areas lie between 10 and 500 mm^2. Thus, the variable bounds are as

$$10 \times 10^{-6} \leq A_1, A_2, A_3, A_4 \leq 500 \times 10^{-6}.$$

In the following, we present the above truss structure problem in NLP form, which is suitable for solving by using an optimization algorithm described in Chapter 4:

$$\text{Minimize} \quad 1.132A_1\ell + 2A_2\ell + 1.789A_3\ell + 1.2A_4\ell$$

subject to

$$S_{yc} - \frac{P}{2A_1 \sin\theta} \geq 0,$$

$$S_{yt} - \frac{P}{2A_2 \cot\theta} \geq 0,$$

$$S_{yt} - \frac{P}{2A_3 \sin\alpha} \geq 0,$$

$$S_{yc} - \frac{P}{2A_4}(\cot\theta + \cot\alpha) \geq 0,$$

$$\frac{\pi E A_1^2}{1.281\ell^2} - \frac{P}{2\sin\theta} \geq 0,$$

$$\frac{\pi E A_4^2}{5.76\ell^2} - \frac{P}{2}(\cot\theta + \cot\alpha) \geq 0,$$

$$\delta_{\max} - \frac{P\ell}{E}\left(\frac{0.566}{A_1} + \frac{0.500}{A_2} + \frac{2.236}{A_3} + \frac{2.700}{A_4}\right) \geq 0,$$

$$10 \times 10^{-6} \leq A_1, A_2, A_3, A_4 \leq 500 \times 10^{-6}.$$

This shows the formulation of the truss structure problem. The seven-bar truss shown in Figure 1.3 is statically determinate and the axial force, stress, and deflection were possible to compute exactly. In cases where the truss is statically indeterminate and large (for hand calculations), the exact computations of stress and deflection may not be possible. A finite element software may be necessary to compute the stress and deflection in any member and at any point in the truss. Although similar constraints can then be formulated with the simulated stresses and deflections, the optimization algorithm which may be used to solve the above seven-bar truss problem may not be efficient to solve the resulting NLP problem for statically indeterminate or large truss problems. The difficulty arises due to the inability to compute the gradients of the constraints. We shall discuss more about this aspect in Chapter 4.

1.2.2 Optimal design of an ammonia reactor

In the ammonia reactor design problem, feed gas containing nitrogen, hydrogen, methane, argon, and a small percentage of ammonia enters the bottom of the reactor (Figure 1.4). Thereafter, the feed gas rises

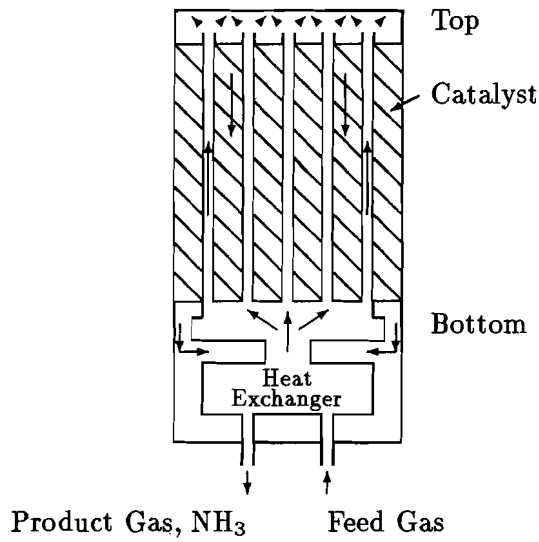


Figure 1.4 A typical ammonia synthesis reactor.

till it reaches the top of the reactor. Then, while moving downward, the nitrogen and hydrogen present in the feed gas undergo reaction to form ammonia in the presence of a catalyst placed in the reactor.

The production of ammonia depends on the temperature of the feed gas, the temperature at the top of the reactor, the partial pressures of the reactants (nitrogen and hydrogen), and the reactor length. The optimal design problem requires to achieve of the optimal reactor length yielding maximum economic returns (profits) from the reactor operation corresponding to various top temperatures.

In this problem, we identify four design variables—the reactor length x, molar flow rate of nitrogen per unit catalyst area N_{N_2}, the feed gas temperature T_f, and the reacting gas temperature T_g. In order to maintain the energy balance of reactions in the reactor, three coupled differential equations must be satisfied (Murase, Roberts, and Converse, 1970; Upreti and Deb, 1994). First, the decrease in the feed gas temperature must be according to the heat loss to the reaction gas:

$$\frac{dT_f}{dx} = -\frac{US_1}{WC_{pf}}(T_g - T_f). \tag{1.2}$$

In Equation (1.2), U is the overall heat transfer coefficient, S_1 is the surface area of the catalyst tubes per unit reactor length, W is the total mass flow rate, and C_{pf} is the specific heat capacity of the feed gas. Secondly, the change in the reaction gas temperature must be according to the heat gain from the feed gas and heat generated in the reaction:

$$\frac{dT_g}{dx} = -\frac{US_1}{WC_{pg}}(T_g - T_f) + \frac{(-\Delta H)S_2}{WC_{pg}} f_a \times \left(K_1 \frac{1.5 p_{N_2} p_{H_2}}{p_{NH_3}} - K_2 \frac{p_{NH_3}}{1.5 p_{H_2}} \right), \tag{1.3}$$

where the parameters K_1 and K_2 are as follows:

$$K_1 = 1.78954 \times 10^4 \exp\left[-20800/(RT_g)\right],$$

$$K_2 = 2.5714 \times 10^{16} \exp\left[-47400/(RT_g)\right].$$

The parameter S_2 denotes the cross-sectional area of the catalyst zone, ΔH is the heat of reaction, C_{pg} is the specific heat capacity of the reacting gas, f_a is the catalyst activity, and R is the ideal gas constant. The parameters p_{N_2}, p_{H_2}, and p_{NH_3} are the partial pressures of nitrogen, hydrogen, and ammonia, respectively. Thirdly, the mass balance of nitrogen yields

$$\frac{dN_{N_2}}{dx} = -f_a \left(K_1 \frac{1.5 p_{N_2} p_{H_2}}{p_{NH_3}} - K_2 \frac{p_{NH_3}}{1.5 p_{H_2}} \right). \tag{1.4}$$

The partial pressures in the above differential equations are computed as

$$p_{N_2} = \frac{286 N_{N_2}}{2.598 N_{N_2} + 2 N_{N_2}},$$

$$p_{H_2} = 3 p_{N_2},$$

$$p_{NH_3} = \frac{286(2.23 N_{N_{2_0}} - 2 N_{N_2})}{2.598 N_{N_{2_0}} + 2 N_{N_2}},$$

where $N_{N_{2_0}}$ is the molar flow rate of nitrogen per unit catalyst area at the top of the reactor. We use the following parameter values:

$$
\begin{aligned}
C_{pf} &= 0.707 \text{ kcal/(kg K)}, \qquad S_1 = 10 \text{ m}, \\
C_{pg} &= 0.719 \text{ kcal/(kg K)}, \qquad S_2 = 0.78 \text{ m}, \\
\Delta H &= -26,600 \text{ kcal/kmol nitrogen}, \qquad R = 1.987 \text{ kcal/(kmol K)}, \\
U &= 500 \text{ kcal/(m}^2 \text{ h K)}, \qquad W = 26,400 \text{ kg/h}, \\
f_a &= 1.
\end{aligned}
$$

Note that all the above three differential equations (1.2)-(1.4) are coupled to each other. In order to solve these equations, we use the following boundary conditions:

$$
\begin{aligned}
T_f(x = 0) &= T_0, \\
T_g(x = 0) &= 694 \text{ K}, \\
N_{N_2}(x = 0) &= 701.2 \text{ kmol/(m}^2 \text{ h)}.
\end{aligned}
$$

The three constraints (Equations (1.2) to (1.4)) can be used to eliminate three of the four design variables and express the optimization problem in terms of only one design variable. We choose to keep the reactor length (x) as the design variable.

The objective of the reactor design problem is to achieve as much profit as possible in the production of ammonia. Considering the net value of ammonia production as the difference between the value of the product gas (heating value and the ammonia value) and the value of feed gas (as a source of heat only) less the amortisation of reactor capital costs, we formulate the profit function as the objective function (in \$/year):

$$
\begin{aligned}
f(x, N_{N_2}, T_f, T_g) = {} & 1.33563 \times 10^7 - 1.70843 \times 10^4 N_{N_2} \\
& + 704.09(T_g - T_0) - 699.27(T_f - T_0) \\
& - [3.45663 \times 10^7 + 1.98365 \times 10^9 x]^{1/2}. \qquad (1.5)
\end{aligned}
$$

Although the above function depends on four variables, three variables (N_{N_2}, T_f, and T_g) are computed from equality constraints. As in the previous example, this problem also requires numerical solution of coupled differential equations. Thus, the NLP problem is as follows:

$$
\text{Maximize} \quad f(x, N_{N_2}, T_f, T_g)
$$

subject to

$$\frac{dT_f}{dx} = -\frac{US_1}{WC_{pf}}(T_g - T_f),$$

$$\frac{dT_g}{dx} = -\frac{US_1}{WC_{pg}}(T_g - T_f) + \frac{(-\Delta H)S_2}{WC_{pg}} f_a \times \left(K_1 \frac{1.5 p_{N_2} p_{H_2}}{p_{NH_3}} - K_2 \frac{p_{NH_3}}{1.5 p_{H_2}}\right),$$

$$\frac{dN_{N_2}}{dx} = -f_a \left(K_1 \frac{1.5 p_{N_2} p_{H_2}}{p_{NH_3}} - K_2 \frac{p_{NH_3}}{1.5 p_{H_2}}\right).$$

1.2.3 Optimal design of a transit schedule

Figure 1.5 shows a typical transit system network. The solid lines

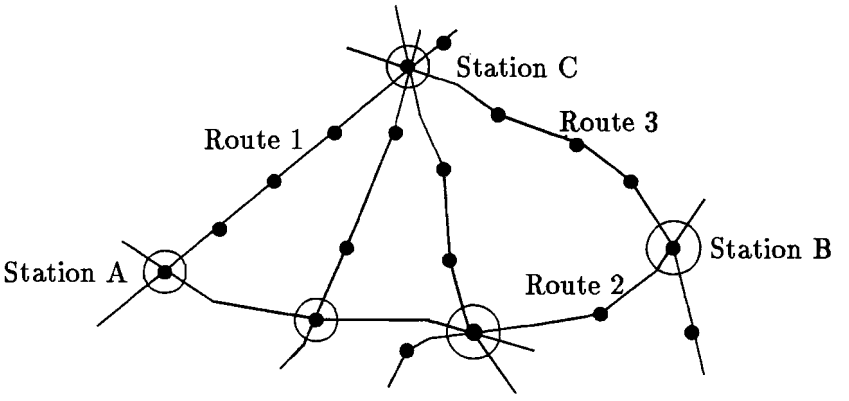


Figure 1.5 A typical transit system network.

represent different routes, the points on the lines represent the stops and the circled intersections of the routes represent the transfer stations. The problem is to determine schedules for the routes such that the transit system provides the best level of service (LOS) to its passengers, within the resources available. One of the good measures of the LOS is the amount of time passengers wait during their journey—the lesser the waiting time, the better is the LOS (Chakroborty, et al., 1994). On any transit network, passengers wait either to board the vehicle at the station of origin or they wait at a transfer station at which they transfer from one vehicle to another. For example, a passenger wishing to travel from station

A to station B (in the network shown in Figure 1.5) will have to wait at station A to board a vehicle on Route 1. Further, the passenger will have to wait at transfer station C to board a vehicle on Route 3 (which will take him/her to the destination). We will refer to the wait at station A as the initial wait time (IWT) and the wait at station C as the transfer time (TT). A good schedule is one which minimizes the sum of IWT and TT for all passengers. Thus, the optimization problem involves finding a schedule of vehicles on all routes (arrival and departure times) such that the total waiting time for the passengers is minimum.

The design variables in this problem are the arrival time a_i^k and departure time d_i^k for the k-th vehicle at i-th route. Thus, if in a problem, there are a total of M routes and each route has K vehicles, the total number of design variables is $2MK$. In addition, there are a few more artificial variables which we shall discuss later.

The constraints in this problem appear from different service related limitations. Some of these constraints are formulated in the following:

Minimum stopping time: A vehicle cannot start as soon as it stops; it has to wait at the stop for a certain period of time, or

$$(d_i^k - a_i^k) \geq s_{\min} \quad \text{for all } i \text{ and } k.$$

Maximum stopping time: A vehicle cannot stop for more than a certain period of time even if it means increasing the total transfer time on the network, or

$$(d_i^k - a_i^k) \leq s_{\max} \quad \text{for all } i \text{ and } k.$$

Maximum allowable transfer time: No passenger on the transit network should have to wait more than a certain period of time T at any transfer station. This can be enforced by checking all possible differences between departure and arrival times and limiting those values to T. Mathematically, this is difficult to achieve. We simplify the formulation of this constraint by introducing a new set of variables $\delta_{i,j}^{k,l}$ between the k-th vehicle of the i-th route and the l-th vehicle of the j-th route. These variables can take either a zero or a one. A value of zero means that the transfer of passengers between those two vehicles is not feasible. A value of one means otherwise. Consider the arrival and departure times of vehicles in two different routes at a particular station, as shown in Figure 1.6. The parame-

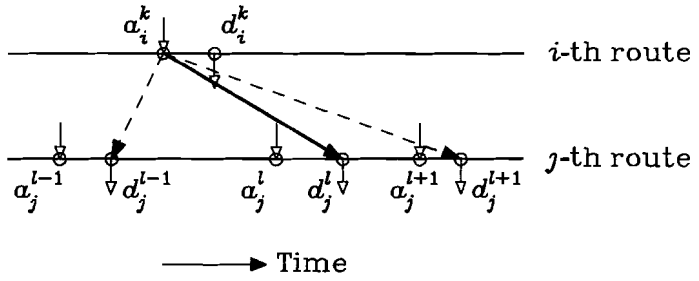


Figure 1.6 Transfers from k-th vehicle on the i-route to three consecutive vehicles in the j-th route.

ters a_i^k and d_i^k are the arrival and departure times of the k-th vehicle in the i-th route. A passenger from the k-th vehicle in the i-th route can only transfer to a vehicle in the j-th route, which is arriving at the station after a_i^k. According to the figure, the transfer of a passenger from the k-th vehicle in the i-th route is not possible to the $(l-1)$-th vehicle in the j-th route, because the departure time of the latter vehicle (d_j^{l-1}) is earlier than a_i^k. Thus, the parameter $\delta_{i,j}^{k,l-1}$ takes a value zero, whereas the parameter $\delta_{i,j}^{k,l}$ takes a value one. In order to simplify the model, we assume that transfers to vehicles departing after l-th vehicle in the j-th route are also not possible. All parameters $\delta_{i,j}^{k,q}$ for $q = (l+1), (l+2), \ldots$ are also zero. Thus, between any two vehicles, the following condition must be satisfied:

$$(d_j^l - a_i^k)\delta_{i,j}^{k,l} \leq T \quad \text{for all } i, j, k \text{ and } l.$$

It is clear that the left side expression of the above condition is zero for those transfers that are not feasible. Since transfers only to the next available vehicle are assumed, only one $\delta_{i,j}^{k,l}$ for $(l = 1, 2, \ldots)$ is one and the rest all are zeros for fixed values of i, j, and k. Mathematically,

$$\sum_l \delta_{i,j}^{k,l} = 1 \quad \text{for all } i, j \text{ and } k.$$

The introduction of the artificial variables $\delta_{i,j}^{k,l}$ makes the formulation easier, but causes a difficulty. Many optimization algorithms cannot handle discrete design variables efficiently. Since the artificial design variables $\delta_{i,j}^{k,l}$ can only take a value zero or one, another set of

constraints is added to enforce the binary values:

$$(d_j^l - a_i^k) + M(1 - \delta_{i,j}^{k,l}) \geq 0 \quad \text{for all } i,\ j,\ k \text{ and } l,$$

where M is a large positive number. The above constraint ensures that the variable $\delta_{i,j}^{k,l}$ always takes a value one whenever a transfer is possible and the value zero whenever transfer is not possible. This constraint is derived purely from the knowledge of the available optimization algorithms. There may be other ways to formulate the concept of feasible transfers, but inclusion of such artificial design variables often make the understanding of the problem easier.

Maximum headway: The headway between two consecutive vehicles should be less than or equal to the policy headway, h_i, or

$$(a_i^{k+1} - a_i^k) \leq h_i \quad \text{for all } i \text{ and } k.$$

The objective function consists of two terms: the first term represents the total transfer time (TT) over all the passengers and the second term represents the initial waiting time (IWT) for all the passengers. The objective is to minimize the following function:

$$\sum_i \sum_j \sum_k \sum_l \delta_{i,j}^{k,l}(d_j^l - a_i^k) w_{i,j}^k$$

$$+ \sum_i \sum_l \int_0^{a_i^k - a_i^{k-1}} v_{i,k}(t)\left[(a_i^k - a_i^{k-1}) - t\right] dt.$$

The parameter $w_{i,j}^k$ is the number of passengers transferring from the k-th vehicle of the i-th route to the j-th route. The first term is obtained by summing the individual transfer time $(d_j^l - a_i^k)$ over all passengers for all the vehicles for every pair of routes. The parameter $v_{i,k}(t)$ is the number of passengers arriving at the stop for the k-th vehicle in the i-th route at a given time t. Since the arrival time for passengers can be anywhere between $t = 0$ to $t = (a_i^k - a_i^{k-1})$ (the headway), the initial waiting time also differs from one passenger to another. For example, a passenger arriving at the stop just after the previous vehicle has left has to wait for the full headway time $(a_i^k - a_i^{k-1})$ before the next vehicle arrives. On the other hand, a passenger arriving at the stop later has to wait for a shorter time. The calculation of the second term assumes that passengers arrive at the stop during the time interval a_i^{k-1} to a_i^k according to the known

time-varying function $v_{i,k}(t)$, where t is measured from a_i^{k-1}. Then the quantity

$$\int_0^{a_i^k - a_i^{k-1}} v_{i,k}(t)\Big[(a_i^k - a_i^{k-1}) - t\Big]\, dt$$

gives the sum of the initial waiting times for all passengers who board the k-th vehicle of the i-th route. We then sum it over all the routes and vehicles to estimate the network total of the IWT. Thus, the complete NLP problem can be written as follows:

$$\text{Minimize} \quad \sum_i \sum_j \sum_k \sum_l \delta_{i,j}^{k,l}(d_j^l - a_i^k) w_{i,j}^k$$

$$+ \sum_i \sum_l \int_0^{a_i^k - a_i^{k-1}} v_{i,k}(t)\Big[(a_i^k - a_i^{k-1}) - t\Big]\, dt$$

subject to

$$\begin{aligned}
& s_{\max} - (d_i^k - a_i^k) \geq 0 && \text{for all } i \text{ and } k,\\
& (d_i^k - a_i^k) - s_{\min} \geq 0 && \text{for all } i \text{ and } k,\\
& T - (d_j^l - a_i^k)\delta_{i,j}^{k,l} \geq 0 && \text{for all } i,\ j,\ k \text{ and } l,\\
& (d_j^l - a_i^k) + M(1 - \delta_{i,j}^{k,l}) \geq 0 && \text{for all } i,\ j,\ k \text{ and } l,\\
& h_i - (a_i^{k+1} - a_i^k) \geq 0 && \text{for all } i \text{ and } k,\\
& \sum_l \delta_{i,j}^{k,l} = 1 && \text{for all } i,\ j \text{ and } k.
\end{aligned}$$

In the above NLP problem, the variables $\delta_{i,j}^{k,l}$ are binary variables taking only a value zero or a one and other variables a_i^k and d_i^k are real-valued. Thus, a mixed integer programming technique described in Chapter 5 or genetic algorithms described in Chapter 6 can be used to solve the above NLP problem (Chakroborty, et al., 1994).

1.2.4 Optimal design of a car suspension

The comfort in riding a car largely depends on the suspension characteristics. The car body is usually supported by a suspension coil spring and a damper at each wheel (Figure 1.7). In some cars, the axle assembly is directly supported on the wheel. The tyre of

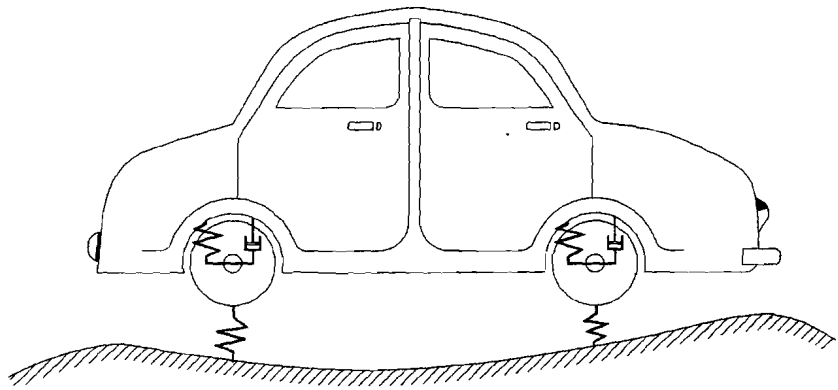


Figure 1.7 A two-dimensional model of a car suspension system.

the wheel can also be assumed to have some stiffness in the vertical direction. A two-dimensional dynamic model of a car suspension system is shown in Figure 1.8. In this model, only two wheels (one each at rear and front) are considered. The sprung mass of the car is considered to be supported on two axles (front and rear) by means of a suspension coil spring and a shock absorber (damper). Each axle contains some unsprung mass which is supported by the tyre.

In order to formulate the optimal design problem, the first task

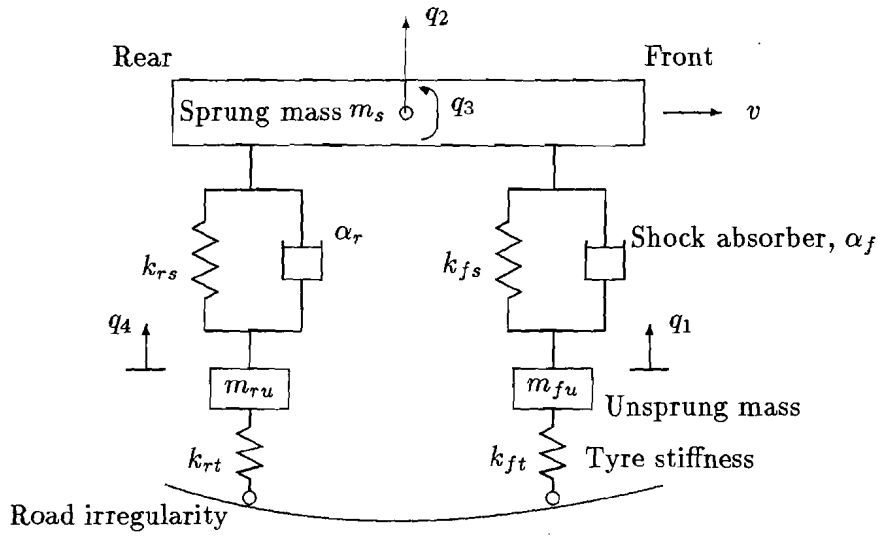


Figure 1.8 The dynamic model of the car suspension system. The above model has four degrees-of-freedom (q_1 to q_4).

is to identify the important design variables. Let us first identify all the design parameters that could govern the dynamic behaviour of the car vibration. In the following, we list all these parameters:

Sprung mass m_s,

Front unsprung mass m_{fu},

Rear unsprung mass m_{ru},

Rear damper coefficient α_r,

Front damper coefficient α_f,

Polar moment of inertia of the car J.

Front coil stiffness k_{fs},

Rear coil stiffness k_{rs},

Front tyre stiffness k_{ft},

Rear tyre stiffness k_{rt},

Axle-to-axle distance ℓ,

We may consider all the above parameters as design variables, but the time taken for convergence of the optimization algorithm may be too much. In order to simplify the formulation, we consider only four of the above parameters—front coil stiffness k_{fs}, rear coil stiffness k_{rs}, front damper coefficient α_f, and rear damper coefficient α_r—as design variables. We keep the other design parameters as constant:

$$m_s = 1000 \text{ kg}, \quad m_{fu} = 70 \text{ kg}, \quad m_{ru} = 150 \text{ kg},$$

$$k_{ft} = 20 \text{ kg/mm}, \quad k_{rt} = 20 \text{ kg/mm}, \quad J = 550 \text{ kg-m}^2,$$

$$\ell_1 = 1.6 \text{ m}, \quad \ell_2 = 1.6 \text{ m}, \quad \ell = 3.2 \text{ m}.$$

The parameters ℓ_1 and ℓ_2 are the horizontal distance of the front and rear axle from the centre of gravity of the sprung mass. Using these parameters, differential equations governing the vertical motion of the unsprung mass at the front axle (q_1), the sprung mass (q_2), and the unsprung mass at the rear axle (q_4), and the angular motion of the sprung mass (q_3) are written (Figure 1.8):

$$\ddot{q}_1 = (F_2 + F_3 - F_1)/m_{fu}, \tag{1.6}$$

$$\ddot{q}_2 = -(F_2 + F_3 + F_4 + F_5)/m_s, \tag{1.7}$$

$$\ddot{q}_3 = [(F_4 + F_5)\ell_2 - (F_2 + F_3)\ell_1]/J, \tag{1.8}$$

$$\ddot{q}_4 = (F_4 + F_5 - F_6)/m_{ru}, \tag{1.9}$$

where the forces F_1 to F_6 are calculated as follows:

$$F_1 = k_{ft} d_1, \qquad F_2 = k_{fs} d_2, \qquad F_3 = \alpha_f \dot{d}_2,$$
$$F_4 = k_{rs} d_4, \qquad F_5 = \alpha_r \dot{d}_4, \qquad F_6 = k_{rt} d_3.$$

The parameters d_1, d_2, d_3, and d_4 are the relative deformations in the front tyre, the front spring, the rear tyre, and the rear spring, respectively. Figure 1.8 shows all the four degrees-of-freedom of the above system (q_1 to q_4). The relative deformations in springs and tyres can be written as follows:

$$d_1 = q_1 - f_1(t),$$
$$d_2 = q_2 + \ell_1 q_3 - q_1,$$
$$d_3 = q_4 - f_2(t),$$
$$d_4 = q_2 - \ell_2 q_3 - q_4.$$

The time-varying functions $f_1(t)$ and $f_2(t)$ are road irregularities as functions of time. Any function can be used for $f_1(t)$. For example, a bump can be modelled as $f_1(t) = A\sin(\pi t/T)$, where A is the amplitude of the bump and T the time required to cross the bump. When a car is moving forward, the front wheel experiences the bump first, while the rear wheel experiences the same bump a little later, depending upon the speed of the car. Thus, the function $f_2(t)$ can be written as $f_2(t) = f_1(t - \ell/v)$, where ℓ is the axle-to-axle distance and v is the speed of the car. For the above bump, $f_2(t) = A\sin(\pi(t - \ell/v)/T)$. The coupled differential equations specified in Equations (1.6) to (1.9) can be solved using a numerical integration technique (for example, a fourth-order Runge-Kutta method can be used) to obtain the pitching and bouncing dynamics of the sprung mass m_s. Equations can be integrated for a time range from zero to $t_{\max}$.

After the design variables are chosen, the next task is to formulate the constraints associated with the above car suspension problem. In order to simplify the problem, we consider only one constraint. The *jerk* (the rate of change of the vertical acceleration of the sprung mass) is a major factor concerning the comfort of the riding passengers. The guideline used in car industries suggests that the maximum jerk experienced by the passengers should not be more than about 18 m/s^3. Mathematically,

$$\max\ q_2'''(t) \leq 18.$$

When the four coupled differential equations (1.6) to (1.9) are solved, the above constraint can be computed by numerically differentiating the vertical movement of the sprung mass (q_2) thrice with respect to time.

The next task is to formulate the objective function. In this problem, the primary objective is to minimize the transimissibility factor which is calculated as the ratio of the bouncing amplitude $q_2(t)$ of the sprung mass to the road excitation amplitude A. Thus, we write the objective function as

$$\text{Minimize} \quad \frac{\max \text{ abs } q_2(t)}{A}.$$

The above objective function can be calculated from the solution of the four differential equations mentioned earlier. A minimum value of the transmissibility factor suggests the minimum transmission of road vibration to the passengers. This factor is also directly related to the *ride* characteristics as specified by the ISO standard. Thus, the optimized design of the above car suspension system would provide the minimum transmissibility of the road vibration to the passengers with a limited level of jerk.

Finally, a minimum and maximum limit for each design variable can be set. This may require some previous experience with a car suspension design, but the following limits for the above car may include the optimal solution:

$$0 \leq k_{fs}, k_{rs} \leq 2 \text{ kg/mm},$$

$$0 \leq \alpha_f, \alpha_r \leq 300 \text{ kg/(m/s)}.$$

Thus, the above optimal car suspension design problem can be written in NLP form as follows:

$$\text{Minimize} \quad \frac{\max \text{ abs } q_2(t)}{A}$$

subject to

$$18 - \max \; q_2'''(t) \geq 0,$$

$$0 \leq k_{fs}, k_{rs} \leq 2,$$

$$0 \leq \alpha_f, \alpha_r \leq 300.$$

1.3 Optimization Algorithms

The above optimization problems reveal the fact that the formulation of engineering design problems could differ from problem to problem. Certain problems involve linear terms for constraints and objective function but certain other problems involve nonlinear terms for them. In some problems, the terms are not explicit functions of the design variables. Unfortunately, there does not exist a single optimization algorithm which will work in all optimization problems equally efficiently. Some algorithms perform better on one problem, but may perform poorly on other problems. That is why the optimization literature contains a large number of algorithms, each suitable to solve a particular type of problem. The choice of a suitable algorithm for an optimization problem is, to a large extent, dependent on the user's experience in solving similar problems. This book provides a number of optimization algorithms used in engineering design activities.

Since the optimization algorithms involve repetitive application of certain procedures, they need to be used with the help of a computer. That is why the algorithms are presented in a step-by-step format so that they can be easily coded. To demonstrate the ease of conversion of the given algorithms into computer codes, most chapters contain a representative working computer code. Further, in order to give a clear understanding of the working of the algorithms, they are hand-simulated on numerical exercise problems. Simulations are performed for two to three iterations following the steps outlined in the algorithm sequentially. Thus, for example, when the algorithm suggests to move from Step 5 to Step 2 in order to carry out a conditional statement, the exercise problem demonstrates this by performing Step 2 after Step 5. For the sake of clarity, the optimization algorithms are classified into a number of groups, which are now briefly discussed.

Single-variable optimization algorithms. Because of their simplicity, single-variable optimization techniques are discussed first. These algorithms provide a good understanding of the properties of the minimum and maximum points in a function and how optimization algorithms work iteratively to find the optimum point in a problem. The algorithms are classified into two categories—direct methods and gradient-based methods. Direct methods do not use any derivative information of the objective function; only objective function values are used to guide the search process. However, gradient-based methods use derivative information (first and/or second-order) to guide the search process. Although engineering op-

timization problems usually contain more than one design variable, single-variable optimization algorithms are mainly used as unidirectional search methods in multivariable optimization algorithms.

Multi-variable optimization algorithms. A number of algorithms for unconstrained, multivariable optimization problems are discussed next. These algorithms demonstrate how the search for the optimum point progresses in multiple dimensions. Depending on whether the gradient information is used or not used, these algorithms are also classified into direct and gradient-based techniques.

Constrained optimization algorithms. Constrained optimization algorithms are described next. These algorithms use the single-variable and multivariable optimization algorithms repeatedly and simultaneously maintain the search effort inside the feasible search region. Since these algorithms are mostly used in engineering optimization problems, the discussion of these algorithms covers most of the material of this book.

Specialized optimization algorithms. There exist a number of structured algorithms, which are ideal for only a certain class of optimization problems. Two of these algorithms—integer programming and geometric programming—are often used in engineering design problems and are discussed. Integer programming methods can solve optimization problems with integer design variables. Geometric programming methods solve optimization problems with objective functions and constraints written in a special form.

There exist quite a few variations of each of the above algorithms. These algorithms are being used in engineering design problems since sixties. Because of their existence and use for quite some years, we call these algorithms as traditional optimization algorithms.

Nontraditional optimization algorithms. There exist a number of other search and optimization algorithms which are comparatively new and are becoming popular in engineering design optimization problems in the recent past. Two such algorithms—genetic algorithms and simulated annealing—are discussed in this book.

We have put together about 34 different optimization algorithms. Over the years, researchers and practitioners have modified these algorithms to suit their problems and to increase the efficiency of the algorithms. However, there exist a few other optimization algorithms—stochastic programming methods and dynamic programming method—which are very different than the above algorithms. Because of the space limitation and occasional

use of these algorithms in engineering design problems, we have not included them in this book. A detailed discussion of these algorithms can be found elsewhere (Rao, 1984).

Many engineering optimization problems contain multiple optimum solutions, among which one or more may be the absolute minimum or maximum solutions. These absolute optimum solutions are known as global optimal solutions and other optimum solutions are known as local optimum solutions. Ideally, we are interested in the global optimal solutions because they correspond to the absolute optimum objective function value. Unfortunately, none of the traditional algorithms are guaranteed to find the global optimal solution, but genetic algorithms and simulated annealing algorithm are found to have a better global perspective than the traditional methods. The global optimality issues are discussed in Chapter 6.

Moreover, when an optimal design problem contains multiple global solutions, designers are not only interested in finding just one global optimum solution, but as many as possible for various reasons. Firstly, a design suitable in one situation may not be valid in another situation. Secondly, it is also not possible to include all aspects of the design in the optimization problem formulation. Thus, there always remains some uncertainty about the obtained optimal solution. Thirdly, designers may not be interested in finding the absolute global solution, instead may be interested in a solution which corresponds to a marginally inferior objective function value but is more amenable to fabrication. Thus, it is always prudent to know about other equally good solutions for later use. However, if the traditional methods are used to find multiple optimal solutions, they need to be applied a number of times, each time starting from a different initial solution and hoping to achieve a different optimal solution each time. Genetic algorithms described in Chapter 6 allow an easier way to find multiple optimal solutions simultaneously in a single simulation.

Another class of optimization problems deals with simultaneous optimization of multiple objective functions. In formulating an optimal design problem, designers are often faced with a number of objective functions. For example, the truss structure problem described earlier should really be reformulated as the minimization of both the weight of the truss and the deflection at the point C. Multiobjective optimization problems give rise to a set of optimal solutions known as *Pareto-optimal* solutions (Chankong and Haimes, 1983), all of which are equally important as far as all objectives are concerned. Thus, the aim in these problems is to find as many Pareto-optimal solutions as possible. Because of the complexity

involved in the multiobjective optimization algorithms, designers usually choose to consider only one objective and formulate other objectives as constraints. Genetic algorithms described in Chapter 6 demonstrate one way to handle multiple objectives and help find multiple Pareto-optimal solutions simultaneously.

At the end of the optimization process, one obvious question may arise: Is the obtained solution a true optimum solution? Unfortunately, there is no easy answer to this question for all optimization problems. In problems where the objective functions and constraints can be written in simple, explicit mathematical forms, the Kuhn-Tucker conditions described in Chapter 4 may be used to check the optimality of the obtained solution. However, those conditions are valid only for a few classes of optimization problems. In a generic problem, this question is answered in a more practical way. In many engineering design problems, a good solution is usually known either from the previous studies or from experience. After formulating the optimal problem and applying the optimization algorithm if a better solution is obtained, the new solution becomes the current best solution. The optimality of the obtained solution is usually confirmed by applying the optimization algorithms a number of times from different initial solutions.

1.4 Summary

In order to use optimization algorithms in engineering design activities, the first task is to formulate the optimization problem. The formulation process begins with identifying the important design variables that can be changed in a design. The other design parameters are usually kept fixed. Thereafter, constraints associated with the design are formulated. The constraints may arise due to resource limitations such as deflection limitations, strength limitations, frequency limitations, and others. Constraints may also arise due to codal restrictions that govern the design. The next task is to formulate the objective function which the designer is interested in minimizing or maximizing. The final task of the formulation phase is to identify some bounding limits for the design variables.

The formulation of an optimization problem can be more difficult than solving the optimization problem. Unfortunately, every optimization problem requires different considerations for formulating objectives, constraints, and variable bounds. Thus, it is not possible to describe all considerations in a single book. However, many of these considerations require some knowledge about the

problem, which is usually available with the experienced designers due to their involvement with similar other design problems.

The rest of the book assumes that the formulation of an optimization problem is available. Chapters 2 to 6 describe a number of different optimization algorithms—traditional and nontraditional—in step-by-step format. To demonstrate the working of each algorithm, hand-simulations on a numerical example problem are illustrated. Sample computer codes for a number of optimization algorithms are also appended to demonstrate the ease of conversion of other algorithms into similar computer codes.

REFERENCES

Chakroborty, P., Deb, K., and Subrahmanyam, P. (1995): Optimal scheduling of urban transit systems using genetic algorithms. *ASCE Journal of Transportation Engineering*, **121**(6), 544–553.

Chankong, V. and Haimes, Y. Y. (1983): *Multiobjective Decision Making Theory and Methodology*. New York: North-Holland.

Murase, A., Roberts, H. L., and Converse, A. O. (1970): Optimal thermal design of an autothermal ammonia synthesis reactor. *Ind. Eng. Chem. Process Des. Develop.*, **9**, 503–513.

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

Shigley, J. E. (1986): *Mechanical Engineering Design*. New York: McGraw-Hill.

Timoshenko, S. (1986): *Strength of Materials, Part 1: Elementary Theory and Problems*. Delhi: CBS Publishers.

Upreti, S. and Deb, K. (in press): Optimal design of an ammonia synthesis reactor using genetic algorithms. *Computers & Chemical Engineering*, (Also avaliable as Technical Report No. IITK/ME/SMD-940015). Kanpur: Department of Mechanical Engineering, Indian Institute of Technology, Kanpur.