Structural Optimization
William R. Spillers · Keith M. MacBain

Structural Optimization

Springer
for our wives
Preface

Contemporary structural optimization has its roots in the 1960s with Lucien Schmidt’s seminal paper.\(^1\) Prior to that time there were no texts on nonlinear programming and if you wanted to do optimization you were relegated to using linear programming. Once mathematical programming was discovered by designers it was thought that engineering design, as an area of study, was over since all you had to do was formulate your design as a nonlinear programming problem and invoke some canned solver. That turned out not to be the case.

While the 1960s and 1970s were characterized by difficulties in solving even small optimization problems (forgetting for the moment optimality criteria methods), the 1990s were characterized by discussions of mathematical programming methods for solving large systems. The flavor of these discussions can be found in the fact that workers in linear programming were then solving large systems (Bixby et al., 1991) with, for example, 12 million variables. In fact, today (2007) the web site of Jacek Gondzio describes solving a nonlinear programming problem with 353 million rows and 1010 million columns.

These capabilities of mathematical programming solvers are part of the driving force behind this text: Surely this technology should offer hope to the structural engineer who must commonly deal with large systems. The other factor driving this work is context: We believe that the use of sequential linear programming together with the use of the incremental equations of structures can serve as a focal point about which a quite general structural optimization solver can be developed.

Then there is the question of the availability of software today. In view of the Solver package included in Microsoft EXCEL, an argument can be made that optimization software is now available to everyone. That package is used in this text along with the IMSL routines available with Digital FORTRAN (now sold as Intel FORTRAN). And clearly, all the work described in this text could have as well been done using Matlab. The reader will find a mix of this software used here. The point is that tools are now available for the solution of optimization problems. This includes some freeware available on the Internet that will be discussed later.

With regard to content, this book includes many computer programs most of which are written in FORTRAN. The excuse for focusing so heavily on computing is that contemporary structural optimization is about computing. Parenthetically, we try in this book to do justice to the history of structural optimization but surely have left things out and apologize to those authors whose work has not been properly referenced here.

---

\(^1\) The history of structural optimization has been developed carefully by Wasiutynski and Brandt (1963).
The attempt in writing this book is to help bring the methods of structural optimization into common usage like those of the finite element method. That is, when the structural engineer sits down to design something, he/she should not only have analysis tools available but should also have optimization tools available. In fact, a good case can be made for including these tools in one package since analysis can always be performed as an optimization problem. The engineer could then, for example, automatically change the structural shape as he/she attempts to satisfy some allowable stress constraints. The analysis/redesign cycle could then simply be replaced by optimization steps.

Historically, there has been a tension between proponents of classical optimization methods who claimed that the users of optimality criteria methods were lacking in theory and the users of heuristic schemes such as optimality criteria methods who at the same time claimed that the classical methods were incapable of solving real (large) structures. In view of the tools now available to the engineer, these arguments can be seen to diminish in importance although the optimality criteria methods still have enormous physical appeal.

We attempt to deal fairly with optimality criteria methods but clearly the focus is on the use of sequential linear programming and the incremental equations of structures that comprise a classical approach. The text begins with an Introduction to simple problems of optimization. Then some available tools are discussed. (This second chapter is more formal than the remainder of the text and the first-time reader might treat it lightly.) Chapter 3 introduces our central topic which is structural optimization approached via the incremental equations of structures and sequential linear programming. Chapter 4 then discusses some problems solved using optimality criteria methods. The remainder of the text offers what we see as an overview of the field of structural optimization. This includes beams and plates, dynamic systems, multicriteria methods, and a brief discussion of some ongoing work. The text closes with an Appendix containing three reprints that are regarded by the authors to be basic to the historical development of structural optimization.

Finally, it has been a pleasure to work Springer Science+Business Media and Steven Elliot our editor. For us, Springer has managed to bring the capabilities and competence of an enormous organization to bear without losing the personal touch of staff contacts. You could not ask for more than that.

Keith M. MacBain
Suffern, NY
William Spillers
Newark, NJ
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>A Frame Example</td>
<td>60</td>
</tr>
<tr>
<td>3.8</td>
<td>A Buckling Example</td>
<td>63</td>
</tr>
<tr>
<td>3.9</td>
<td>The Incremental Equations when Shape Change is Allowed</td>
<td>67</td>
</tr>
<tr>
<td>3.10</td>
<td>A Beam Example</td>
<td>72</td>
</tr>
<tr>
<td>3.11</td>
<td>A Plate Bending Problem</td>
<td>74</td>
</tr>
<tr>
<td>3.12</td>
<td>Problems</td>
<td>76</td>
</tr>
<tr>
<td>4.</td>
<td>Optimality Criteria Methods</td>
<td>77</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>77</td>
</tr>
<tr>
<td>4.2</td>
<td>The Most Simple Optimality Criteria Problem</td>
<td>78</td>
</tr>
<tr>
<td>4.3</td>
<td>Monotone Behavior</td>
<td>79</td>
</tr>
<tr>
<td>4.4</td>
<td>An Application</td>
<td>80</td>
</tr>
<tr>
<td>4.5</td>
<td>Sandwich Beams</td>
<td>84</td>
</tr>
<tr>
<td>4.6</td>
<td>A Generalization of the Truss problem</td>
<td>84</td>
</tr>
<tr>
<td>4.7</td>
<td>Plastic Design of Frames</td>
<td>86</td>
</tr>
<tr>
<td>4.8</td>
<td>Sandwich Plate Design</td>
<td>88</td>
</tr>
<tr>
<td>4.9</td>
<td>Truss Design</td>
<td>89</td>
</tr>
<tr>
<td>4.10</td>
<td>A Plane Stress Problem</td>
<td>91</td>
</tr>
<tr>
<td>4.11</td>
<td>Prager and His Co-workers</td>
<td>95</td>
</tr>
<tr>
<td>4.11.1</td>
<td>The Axisymmetric Circular Plate</td>
<td>95</td>
</tr>
<tr>
<td>4.12</td>
<td>A Plate Problem</td>
<td>97</td>
</tr>
<tr>
<td>4.13</td>
<td>Problems</td>
<td>100</td>
</tr>
<tr>
<td>5.</td>
<td>Some Basic Optimization Problems</td>
<td>103</td>
</tr>
<tr>
<td>5.1</td>
<td>Multiple Loading Conditions</td>
<td>103</td>
</tr>
<tr>
<td>5.1.1</td>
<td>An Example</td>
<td>105</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Realizability</td>
<td>106</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Three Loading Conditions</td>
<td>108</td>
</tr>
<tr>
<td>5.2</td>
<td>Deflection Constraints</td>
<td>109</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Funaro’s Example</td>
<td>112</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Optimality Criteria Approach</td>
<td>117</td>
</tr>
<tr>
<td>5.2.3</td>
<td>An Example</td>
<td>121</td>
</tr>
<tr>
<td>5.3</td>
<td>Optimal Shape</td>
<td>122</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Frame Problems</td>
<td>124</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Code Design for a Truss Problem</td>
<td>126</td>
</tr>
<tr>
<td>5.4</td>
<td>Generating New Designs Automatically</td>
<td>128</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Algebraic Methods</td>
<td>130</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Linguistic Methods</td>
<td>135</td>
</tr>
<tr>
<td>5.5</td>
<td>Problems</td>
<td>137</td>
</tr>
<tr>
<td>6.</td>
<td>Beams and Plates: The Work of Rozvany</td>
<td>139</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>139</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Examples</td>
<td>140</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Reaction Costs</td>
<td>143</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Optimal Segmentation</td>
<td>144</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.1.4</td>
<td>A More Realistic Beam Model</td>
<td>145</td>
</tr>
<tr>
<td>6.2</td>
<td>Design of Plates</td>
<td>146</td>
</tr>
<tr>
<td>6.3</td>
<td>Problems</td>
<td>149</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>151</td>
</tr>
<tr>
<td>7.2.1</td>
<td>The Static Case</td>
<td>154</td>
</tr>
<tr>
<td>7.2.2</td>
<td>The Dynamic Case</td>
<td>154</td>
</tr>
<tr>
<td>7.2.3.1</td>
<td>Tuned Mass Dampers</td>
<td>160</td>
</tr>
<tr>
<td>7.3.1</td>
<td>A Truss Problem</td>
<td>165</td>
</tr>
<tr>
<td>7.3.2.1</td>
<td>Tuned Mass Dampers</td>
<td>161</td>
</tr>
<tr>
<td>7.3.3.1</td>
<td>Tuned Mass Dampers</td>
<td>166</td>
</tr>
<tr>
<td>7.3.3</td>
<td>An Incremental Solution of the Truss Problem</td>
<td>171</td>
</tr>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>175</td>
</tr>
<tr>
<td>8.2</td>
<td>Solving Multicriteria Optimization Problems</td>
<td>177</td>
</tr>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>179</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Effect of Fabrication Costs</td>
<td>181</td>
</tr>
<tr>
<td>9.3.1</td>
<td>The Effect of Shape</td>
<td>184</td>
</tr>
<tr>
<td>9.4</td>
<td>Problems</td>
<td>186</td>
</tr>
<tr>
<td>10.1</td>
<td>Design of Tall Buildings</td>
<td>187</td>
</tr>
<tr>
<td>10.1.1</td>
<td>Wind Loads on Tall Buildings</td>
<td>187</td>
</tr>
<tr>
<td>10.1.2</td>
<td>Tuned Mass Dampers</td>
<td>188</td>
</tr>
<tr>
<td>10.1.3</td>
<td>Stochastic Processes</td>
<td>189</td>
</tr>
<tr>
<td>10.2</td>
<td>Heuristic Algorithms</td>
<td>192</td>
</tr>
<tr>
<td>10.3</td>
<td>Extending the Design Process</td>
<td>193</td>
</tr>
<tr>
<td>10.4</td>
<td>Design Theory</td>
<td>193</td>
</tr>
<tr>
<td>10.4.1</td>
<td>Robust Design</td>
<td>194</td>
</tr>
<tr>
<td>10.4.2</td>
<td>Creativity in Design</td>
<td>194</td>
</tr>
<tr>
<td>10.4.3</td>
<td>Design Ontologies</td>
<td>195</td>
</tr>
<tr>
<td>10.4.4</td>
<td>Architectural Design</td>
<td>195</td>
</tr>
<tr>
<td>10.5</td>
<td>Available Computational Algorithms</td>
<td>195</td>
</tr>
<tr>
<td>A.1</td>
<td>Using Computer Languages and Programs</td>
<td>197</td>
</tr>
<tr>
<td>A1.1</td>
<td>Fortran</td>
<td>197</td>
</tr>
</tbody>
</table>
B. The Node Method for Trusses ................................................................. 207
   B.1 Introduction ..................................................................................... 207
   B.2 A Formal Description of the Truss Problem .................................... 208
   B.3 A Decomposition ............................................................................ 212

C. Convex Sets and Functions: Homogeneous Functions ...................... 219
   C.1 Convex Sets and Functions ............................................................. 219
   C.2 Homogeneous Functions .................................................................. 220

D. Structural Optimization Classics ......................................................... 223
   D.1 Michell Trusses .............................................................................. 223
   D.2 Keller’s Optimal Column ................................................................. 233
   D.3 The Paper of Venkayya, Khot, and Berke ....................................... 245

References .................................................................................................. 289

Index .......................................................................................................... 299
Contents of the CD

Readme

The readme.txt file contains some basic information as to how to use the CD. More detailed information is given in the text and in the Tools folder.

Computer Programs

This folder contains source code for all programs discussed in the text. A list of these programs is given below.

List of Programs

Program 1. (Prog01.xls) This Excel worksheet is used in Chapter 1 to solve the two-bar truss problem.
Program 2. (Prog02.xls) This Excel worksheet is used in Chapter 1 to solve the frame design.
Program 3. (Prog03.for) This FORTRAN program is used in Section 2.6 of the text as an example of the sequential linear programming solver.
Program 4. (Prog04.xls) This Excel worksheet uses Solver check of the problem solved by Program 3.
Program 5. (Prog05.for) This FORTRAN program is used in Appendix C as a sequential linear programming solver.
Program 6. (Prog06.xls) This Excel worksheet is the Solver solution used in Appendix C.
Program 7. (Prog07.m) This Word file contains a Matlab script file and a Matlab function. They must be separated before they are used. This program is a Matlab version of sequential linear programming.
Program 8. (Prog08.xls) This Excel worksheet is a study of the use of visual basic for structural optimization.
Program 9. (Prog09.for) This FORTRAN program solves a truss with a displacement constraint from Section 3.4 in the text. It is to be used with the data file Prog09.dat.
Program 10. (Prog10.for) This is Program 9 with stress constraints added. It is to be used with the data file Prog10.dat.
Program 11. (Prog11.for) This FORTRAN program solves the classic 25-bar truss problem. It is to be used with the data file Prog11.dat.
Program 12. (Prog12.for) This FORTRAN program solves a plane frame problem with a displacement constraint as discussed in Section 3.7. It is to be used with the data file Prog12.dat.
Program 13. (Prog13.for) This FORTRAN program solves Keller’s problem of the optimal shape for a column.

Program 14. (Prog14.for) This FORTRAN program solves the geometric optimization problem discussed in Section 3.9 in the text. It is to be used with the data file Prog14.dat.

Program 15. (Prog15.for and Prog15.dat) This is another version of Program 14.

Program 16. (Prog16.m) This Matlab program solves the problem of an optimal fixed beam as discussed in Section 3.10.

Program 17. (Prog17.m) This Matlab program solves the problem of an optimal fixed plate as discussed in Section 3.11.

Program 18. (Prog18.for) This FORTRAN program solves the most simple iterative design problem for trusses discussed in Section 4.4 of the text. It uses Prog18.dat as a data file.

Program 19. (Prog19.for) This FORTRAN program solves the iterative design problem for plane frames discussed in Section 4.7 of the text. It uses Prog19.dat as a data set.

Program 20. Program 20 comprises three programs: Prog20.for, a basic finite element program; Prog21.for, a mesh generator; and Prog22.for, an iterative design program for plane stress problems. The use of these programs is discussed in Section 4.10 in the text.

Program 23. (Prog23.for) This FORTRAN program solves the iterative design problem for a fixed plate as discussed in Section 4.12 of the text.

Program 24. (Prog24.m and Prog25.m) These Matlab programs comprise a program for the solution of the transient response of a tapered beam as discussed in Section 7.2 of the text.

Program 26. (Prog26.for) This FORTRAN program solves the problem of the optimal steady-state vibrations of a truss as discussed in Section 7.3 of the text. It is run with Prog26.dat as a data file.

Program 27. (Prog27.for) This is an incremental solution of the problem solved in Program 26. It uses the same data file.

Program 28. (Prog28.for) This FORTRAN program resolves an example of Farkas from Section 9.3 using AISC stresses. It is to be run with the data file Prog28.dat.

Program 29. (Prog29.for) This FORTRAN program solves the three-dimensional truss. It is used in Problem 3 of Chapter 1. It is to be run with the data file Prog29.dat.

Program 30. (Prog30.m) This Matlab program executes the simplex method. It is an educational program not intended for serious applications.

Program 31. (Prog31.for) This FORTRAN program executes the primal interior point linear programming algorithm described by Arbel. It is to be run with Prog31.dat as a data file.
Program 32. (Prog32.for) This FORTRAN program executes the dual interior point linear programming algorithm described by Arbel. It is to be run with Prog32.dat as a data file.

Program 33. (Prog33.xls) This Excel worksheet is used in Section 9.3 to compute the properties of hollow sections.

Program 34. (Prog34.xls) This Excel worksheet is used in Section 9.3 to compute the properties of angles.

Tools

This folder contains some tools that are considered useful. A list of these tools is given below.

List of Tools

- tr2d: This folder contains an example of using VB code in Excel to automate tasks and using the solver.
- g77.zip: A freely available FORTRAN compiler.
- MinGW: A freely available FORTRAN compiler. This is the compiler that was tested with the files in optools.zip.
- optools: Files that contain IMSL routines used in the text for users without this library.
- unixUtils: Unix-like utilities for a PC.
- unzip: Executable file to unzip files on the disk.
- View3d: Java-based graphical user interface (GUI) framework.
1 Introduction

This chapter introduces the reader to various problems that are typical of the field of optimization. It begins with a discussion of structural optimization described formally as a *mathematical programming problem*. This is followed by a series of *typical* optimization problems and an introduction to some of the tools available to deal with them including Lagrange multipliers and linear programming.

1.1 Problem Statement

Structural optimization problems can be deceptively simple to formulate. They can be written as

Find $x$ to minimize $f(x)$ subject to $g(x) \leq 0$  \hspace{1cm} (1)

Here $f$ (the *objective function*) is a scalar, $x$ is an $n$-vector (has $n$ components), and $g$ (the *constraints*) is an $m$-vector. Problems of this type are called *mathematical programming problems* (Luenberger 1984). Equation (1) is typically simplified to read

$$
\begin{align*}
\min f(x) & \text{ subject to } g(x) \leq 0 \\
& \text{ or even } \min f(x) | g(x) \leq 0
\end{align*}
$$

Note that

$$
g(x) \leq 0 \Rightarrow \begin{cases} 
g_1 & \leq 0 \\
g_2 & \leq 0 \\
\vdots & \\
g_m & \leq 0
\end{cases}
$$

The specific form of this optimization problem statement is not important since minimize $f$ $\iff$ maximize $-f$ and $g(x) \leq 0$ $\iff$ $-g(x) \geq 0$ (see Fig. 1.1).
1.2 An Optimization Problem

Optimization terminology can be difficult for those unfamiliar with it. This chapter presents some common optimization problems and approaches in an attempt to smooth the entrance of the engineer into this world of optimization. The following example will introduce some of the features of mathematical programming problems (Fox 1969).

In this case (Fig. 1.2) the height $H$ and the diameter $d$ of the (tubular) members of a plane two-bar truss are varied in order to minimize the volume of material (which is proportional to the total weight). Additionally, there are two require-
ments (constraints) that are to be satisfied: (1) the member stress should be less than the yield stress, $F_y$, and (2) the members should not buckle. Parameters for this problem are tabulated below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>Young’s modulus</td>
<td>29,000 ksi</td>
</tr>
<tr>
<td>$B$</td>
<td>Half-distance between supports</td>
<td>100 in.</td>
</tr>
<tr>
<td>$F_y$</td>
<td>Yield stress of material</td>
<td>36 ksi</td>
</tr>
<tr>
<td>$t$</td>
<td>Wall thickness of tube</td>
<td>0.25 in.</td>
</tr>
<tr>
<td>$P$</td>
<td>Applied load</td>
<td>100 k</td>
</tr>
</tbody>
</table>

Given this problem statement, the objective function and the constraints can be expressed in a mathematical form. Using elementary analysis, the dependent parameters are tabulated below:

<table>
<thead>
<tr>
<th>Item</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second moment of inertia (in.$^4$)</td>
<td>$I = \frac{\pi}{64}[(d + t)^4 - (d - t)^4]$</td>
</tr>
<tr>
<td></td>
<td>$= \frac{\pi t d}{8}(d^2 + t^2)$</td>
</tr>
<tr>
<td>Member force (k)</td>
<td>$F = \frac{P \sqrt{B^2 + H^2}}{2H}$</td>
</tr>
<tr>
<td>Member stress (ksi)</td>
<td>$\sigma = \frac{F}{A}$</td>
</tr>
<tr>
<td>Buckling stress (ksi)</td>
<td>$\sigma_{cr} = \frac{\pi^2 EI}{L^2} \frac{1}{A}$</td>
</tr>
</tbody>
</table>

The volume of material is simply the bar area, $A$, multiplied by the total length of two bars. Using the approximation $A = \pi t d$ (valid for thin-wall tubes), the volume is simply

$$V = 2AL$$

$$= 2(\pi t d)\sqrt{H^2 + B^2}$$

Formally, the objective function, $f$, and the constraints, $g$, can now be written as
It is worth emphasizing at this point that all optimization problems in this text will be expressed in a similar manner. That is, there is a scalar function $f$ to be optimized subject to zero or more constraints, $g$.

Figure 1.3 is instructive. It shows a plot of the two constraint surfaces (curves) and curves of constant weight. The curves of constant weight are sometimes called level curves and the figure may be viewed as a topographic map of the solution space.

In this case, the solution is found by starting in the upper right area at a feasible point and moving as far “downhill” as possible without passing the “fence” of the constraints.

Using the numerical values given above, this problem can be written as find $d$ and $H$ to
This problem has a unique solution (not all problems do), which may be found by several methods, many of which will be explored in this text. In this case, the “Solver” from Microsoft Excel was used (see Fig. 1.4.) The solution to this problem is at $H = 56$ in., $d = 3.6$ in. More detail on this example and on the use of the Excel Solver can be found in Appendix A and later in this text.

Fig. 1.4. Excel Spreadsheet for Two-Bar Truss Optimization

This spreadsheet is included on the disc as Program 1.

1.3 Elementary Calculus

The engineer typically first encounters optimization in calculus where the derivative is used to identify a stationary point. The second derivative is, of course, required to determine whether a stationary point is a maximum or a minimum. For example (Fig. 1.5), the moment in a uniformly loaded simply supported beam is given by

\[
\begin{align*}
\text{minimize} \quad & f = \frac{\pi}{2} d \sqrt{H^2 + 100^2} \\
\text{subject to} \quad & 50 \frac{\sqrt{H^2 + 100^2}}{H} \frac{4}{d \pi} - 36 \leq 0 \\
& 50 \frac{\sqrt{H^2 + 100^2}}{H} \frac{4}{d \pi} - 35,777 \left( \frac{d^2 + 0.25^2}{H^2 + 100^2} \right) \leq 0
\end{align*}
\]
\[ M(x) = \frac{wx}{2} - \frac{wx^2}{2} \]

Setting the first derivative equal to zero gives a stationary value as

\[
\frac{dM}{dx} = M' = \frac{wL}{2} - wx = 0 \quad \Rightarrow \quad x = \frac{L}{2}
\]

Since the second derivative is negative,

\[
\frac{d^2M}{dx^2} = M'' = -w < 0
\]

the stationary value is a maximum.

Figure 1.5 also shows another case of a loaded beam. Here the moment diagram is segmentally linear which implies that it is no longer possible to differentiate to obtain the maximum or minimum values. This situation is typical of linear programming as seen below.

![Fig. 1.5. Moment Diagrams](image)

### 1.4 Optimal Slope for Truss Bars

Given a two-bar truss (Fig. 1.2), it is possible to ask for the optimal bar slope \( \theta = \tan^{-1} \frac{H}{B} \), this time using only a single stress constraint. Using the same notation given earlier with \( H \) as the (only) independent variable, the elementary calculus approach just presented can be used to solve for \( H \) that minimizes the volume. The total volume of the structure is expressed as
\[
V = 2AL \\
= 2 \left( \frac{F}{\sigma_a} \right) \sqrt{H^2 + B^2} \\
= 2 \left( \frac{P}{2} \frac{H^2 + B^2}{H} \frac{1}{\sigma_a} \right) \sqrt{H^2 + B^2} \\
= \frac{P}{\sigma_a} \frac{H^2 + B^2}{H}
\]

The first derivative with respect to the height \( H \) then gives

\[
\frac{dV}{dH} = \frac{P}{\sigma_a} \frac{H^2 - B^2}{H^2}
\]

and the stationary point is found by setting the first derivative equal to zero.

\[
\frac{dV}{dH} = 0 \quad \Rightarrow \quad H = B
\]

As before, the sign of the second derivative indicates the type of stationary value, in this case it is positive indicating that the solution is indeed a minimum. Additionally, referring again to Fig. 1.2, \( H = B \Rightarrow \theta = 45^\circ \). This example restates what structural engineers know: that truss bars with a flat slope are generally not efficient.

### 1.5 An Arch Problem

If a parabolic cable under uniform load is inverted, the shape formed is a funicular arch (Fig. 1.6). The following problem asks for the optimal height when the design is done under conditions of a constant allowable stress \( \sigma \). Using the geometry shown in the figure, \( y = 4hx^3/L^2 \) \( \Rightarrow \) \( y' = 8hx/L^2 \).
From symmetry, the vertical reactions are \( wL/2 \). A free body diagram of \( \frac{1}{2} \) the arch can be used to compute the horizontal reaction \( H = wL/8h \). If a cut is made at any point \( x \), the vertical force component on the cross section can be determined to be \( wx \). The resultant force (axial) at any point \( x \) is then \( F = \left[ H^2 + (wx)^2 \right]^{1/2} \) and the area \( A = F/\sigma \). The volume of material is then

\[
V = \int_{0}^{L} Adx = 2 \int_{0}^{L/2} \frac{1}{\sigma} \left[ \left( \frac{wL^2}{8h} \right)^2 + (wx)^2 \right]^{1/2} \left[ 1 + \left( \frac{8hx}{L^2} \right)^2 \right]^{1/2} dx
\]

Since the arc length \( ds = [1 + (y')^2]^{1/2} \) \( dx \), the explanation of the integral is complete. With some rearranging of terms, the square roots cancel and the volume is found to be

\[
V = 2w \frac{L^3}{\sigma} \left( \frac{L^3}{16h} + \frac{hL}{3} \right)
\]

Setting \( dV/dh = 0 \) gives \( h_{opt} = \sqrt{3} L/4 \) (Haftka, et al. 1990).

### 1.6 The Gradient of a Function

Given a scalar-valued function of an n-vector \( \mathbf{x} \), say \( f(\mathbf{x}) \), \( f \) is said to define a surface. The gradient, \( \nabla f \), of this scalar function is the vector
The gradient commonly appears in optimization. It can be used, for example, to describe the variation $df$ of a function as

$$df = \nabla f^T \, dx = \nabla f \cdot dx \quad \text{(using a hybrid notation)}$$

with

$$dx = \begin{bmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_n \end{bmatrix}$$

This is just the chain rule of partial differentiation since

$$\nabla f \cdot dx = \frac{\partial f}{\partial x_1} \, dx_1 + \frac{\partial f}{\partial x_2} \, dx_2 + \cdots + \frac{\partial f}{\partial x_n} \, dx_n$$

It follows that if $n$ is a unit vector, the directional derivative (in the direction of $n$) is

$$\frac{\partial f}{\partial n} = \nabla f \cdot n$$

Also, $\nabla f$ is normal to surfaces of constant $f$ since

$$df = \nabla f \cdot dx = 0 \quad \Rightarrow \quad \nabla f \perp dx$$

$\nabla f$ also points uphill in the direction of steepest ascent ($\nabla f \cdot n$ is maximum when $\nabla f$ parallels $n$). For example, let $f = x^2 + y^2 + z^2$. The gradient of $f$ is then
Contours of $f$ are given in Fig. 1.7 along with the gradient at a point and an arbitrary constraint.

![Gradient of a Function](image)

An unconstrained optimization problem can be written using the gradient as $\nabla f = 0$. For the case of $f$ described above, this gives the minimum of $f$ at the point $x = y = z = 0$.

Later the gradient of a vector will also be used. For example, consider a problem with equality constraints, $g(x) = 0$. At some point $x$ which satisfies these constraints there should be a small region $dx$ which also satisfies the constraints. It follows that in this region $\nabla g \cdot dx = 0$. The gradient of a vector is a matrix that must be interpreted so that its components represent the chain rule of partial differentiation in the same manner as the scalar case so that $dg = \nabla g \cdot dx$ can be interpreted again as an application of the chain rule of partial differentiation

$$\nabla g = \begin{bmatrix} \partial g_1 / \partial x_1 & \partial g_1 / \partial x_2 & \ldots & \partial g_1 / \partial x_n \\ \partial g_2 / \partial x_1 & \partial g_2 / \partial x_2 & \ldots & \partial g_2 / \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial g_m / \partial x_1 & \partial g_m / \partial x_2 & \ldots & \partial g_m / \partial x_n \end{bmatrix}$$
1.7 The Lagrange Multiplier Rule

Lagrange multipliers are a basic tool of optimization and will receive considerable attention in this text. However, for the moment it is convenient to follow the approach of Kaplan (1953) in advanced calculus who argues informally that the gradient of the objective function and the gradient of the constraints must be parallel at an optimal point (see Fig. 1.8). He looks at the problem of

\[ \text{max } f = xy \quad \text{subject to} \quad g = x^2 + y^2 - 1 = 0 \]

This problem can be solved using the so-called *Lagrange multiplier rule* in which the Lagrangian \( L \) is formed as

\[ L = f + \lambda \cdot g = xy + \lambda \left( x^2 + y^2 - 1 \right) \]

Here \( L \) is simply the sum of the objective function plus the constraint multiplied by the constant \( \lambda \). (\( \lambda \) is called a Lagrange multiplier.) In general, \( \lambda \) will be a vector with \( m \) terms corresponding to each of the \( m \) constraints of the problem. The optimal solution is then found by differentiating with respect to the variables \( x, y, \) and \( \lambda \) as

\[
\begin{align*}
\frac{\partial L}{\partial x} &= y + 2 \lambda x \\
\frac{\partial L}{\partial y} &= x + 2 \lambda y \\
\frac{\partial L}{\partial \lambda} &= x^2 + y^2
\end{align*}
\]

and setting each of these equations equal to zero. Regardless of the value of \( \lambda \), it follows that \( x^2 = y^2 \), which gives the four points shown in Fig. 1.8. More consideration will be given to the details in Section 2, particularly with regard to the sign of \( \lambda \) and the character of the problem (i.e., maximization or minimization); however, at this point we note that \( x = y = \pm 1 \) are solutions to this problem because they do indeed maximize \( f \).
In general it will be seen later that the Lagrange multiplier method converts an optimization problem with *equality constraints* into a system of nonlinear equations. It can be seen in the example above that the solution of the nonlinear system can lead to some spurious results so that the analyst must be careful when interpreting his/her results.

### 1.8 Newton’s Method

One of the first topics of numerical analysis is Newton’s method. It is generally discussed about the problem of finding roots of an equation such as \( f = 0 \) but as noted in Sections 1.3 and 1.7, optimization problems are often solved by finding the root to some equation (e.g., \( y' = 0 \)). For example, the buckling load of a column that is fixed at one end and pinned at the other (Timoshenko 1936) requires the solution of the equation \( f = x - \tan(x) = 0 \) shown in Fig. 1.9.
That can be done using Newton’s method, which is an iterative scheme based on
the first two terms of a (linearized) Taylor series to give an estimate of the change
in the current estimate, \( dx \)

\[
f(x) \approx f(x_0) + f'(x_0)dx
\]

\[
dx = -\frac{f(x_0)}{f'(x_0)}
\]

More details on Newton’s method is found in Chapter 2; however, here we simply
present the iterative equation

\[
x_{i+1} = x_i + dx
\]

\[
x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}
\]

where one would typically iterate until the term \( f_i \) approaches zero (i.e., the solu-
tion is reached). For this example, \( f' = 1 - \sec^2(x) \). Starting at \( x = 4.4 \), the solution
proceeds as indicated below:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( x-\tan(x) )</th>
<th>( 1-\sec^2(x) )</th>
<th>( dx )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>1.3037</td>
<td>-9.5872</td>
<td>0.1360</td>
</tr>
<tr>
<td>4.5360</td>
<td>-1.0738</td>
<td>-31.4692</td>
<td>-0.0341</td>
</tr>
<tr>
<td>4.5019</td>
<td>-0.1777</td>
<td>-21.8982</td>
<td>-0.0081</td>
</tr>
<tr>
<td>4.4937</td>
<td>-0.0068</td>
<td>-20.2548</td>
<td>-0.0003</td>
</tr>
<tr>
<td>4.4934</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice that in this case the starting solution is rather close to Timoshenko’s solu-
tion of \( x = 4.493 \). That is due to the fact that this problem is rather singular (i.e.,
asymptotic). In other words, the solution is dependent on the initial solution and
an initial value of, say, \( x = 8 \), would give a different solution for this problem (ver-
ify solution to be \( x = 10.904 \). Newton’s method will be used heavily in this text,
particularly in the so-called method of sequential linear programming described in Chapter 2. This example is included in the Excel file “Chapter 1.xls”.

1.9 Solving Linear Equations

Given a system of linear equations, $A \mathbf{x} = \mathbf{b}$, there are three possibilities: a unique solution, many solutions, or no solution. For the case of no solution it is possible to select the vector $\mathbf{x}$ so that the norm of the error, $\mathbf{e}$, is minimized. Let $\mathbf{e} \equiv \mathbf{b} - A\mathbf{x}$ and the norm of $\mathbf{e}$ be defined to be $\|\mathbf{e}\| = (\mathbf{e}^T \mathbf{e})^{1/2}$. Then

$$e^T e = (b - Ax)^T (b - Ax)$$
$$= b^T b - x^T A^T b - b^T Ax + x^T A^T A x$$

Differentiating with respect to $\mathbf{x}$ to find a stationary point gives

$$\frac{\partial (e^T e)}{\partial \mathbf{x}} = 0 \Rightarrow A^T Ax = A^T b \Rightarrow \mathbf{x} = (A^T A)^{-1} A^T b$$

The matrix $(A^T A)^{-1} A^T$ is sometimes called the generalized inverse of the matrix $A$.

There are, of course, many different norms that can be used to measure error. For example, it is common in structures to minimize error using an energy or matrix norm with $\|\mathbf{e}\| \equiv (\mathbf{e}^T K \mathbf{e})^{1/2}$ and $K$, a positive-definite matrix. It follows that in this case

$$A^T K A \mathbf{x} = A^T K b \Rightarrow \mathbf{x} = (A^T K A)^{-1} A^T K b$$

1.10 Linear Systems Versus Optimization

It is frequently possible to replace a linear system by an optimization problem. For example, given a set of linear equations $A\mathbf{x} = \mathbf{b}$, with the matrix $A$ positive definite, an equivalent optimization problem is

$$\text{minimize } \Phi(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{x}^T \mathbf{b}$$

To see this, consider the exact solution $\mathbf{x}^*$. Let $\mathbf{x}^* | A \mathbf{x}^* = \mathbf{b}$ then

$$\mathbf{x}^T A \mathbf{x}^* = \mathbf{x}^T \mathbf{b}$$
Now consider \( x = \hat{x} + \varepsilon \), which differs from the exact solution, \( \hat{x} \), by some small amount \( \varepsilon \). The objective function \( \Phi \) evaluated at \( x \) is then

\[
\Phi(x) = \frac{1}{2} (x + \varepsilon)^T A (x + \varepsilon) - (\hat{x} + \varepsilon)^T b \\
= \frac{1}{2} (x^*)^T A (x^*) + \frac{1}{2} (\varepsilon)^T A (\varepsilon) + (\varepsilon)^T A (x^*) - (x^*)^T b - (\varepsilon)^T b \\
= \Phi(x^*) + \text{positive quantity}
\]

Since \( \varepsilon^T (Ax - b) = 0 \) It follows that

\[
\text{minimize } \Phi(x) = \frac{1}{2} x^T Ax - x^T b \quad \iff \quad Ax = b
\]

### 1.11 Equations of Structures

It is convenient here to introduce the equations of discrete structures as they are to be used in this book. They can be contrasted with the equations of elasticity as indicated.

<table>
<thead>
<tr>
<th>Structures</th>
<th>Elasticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equilibrium equation: ( N^T F = P )</td>
<td>( \sigma_{ij,j} + f_j = 0 )</td>
</tr>
<tr>
<td>Constitutive equation: ( F = K \Delta )</td>
<td>( \sigma_{ij} = 2\mu \varepsilon_{ij} + \lambda \delta_{ij} \varepsilon_{kk} )</td>
</tr>
<tr>
<td>Member/node displacement eq. (strain/displacement) ( \Delta = N \delta )</td>
<td>( \varepsilon_{ij} = \frac{1}{2} (u_i,j + u_j,i) )</td>
</tr>
</tbody>
</table>

Here

- \( F, \Delta (\sigma_{ij}, \varepsilon_{ij}) \) – member force, displacement matrix (stress, strain tensor)
- \( P, \delta (f, u) \) – node force, displacement matrix (traction, displ. vector)
- \( K, (\mu, \lambda) \) – primitive stiffness matrix (Lame constants)
- \( N \) – generalized incidence matrix

As an example of a structural optimization problem, Fig. 1.10 shows a three-bar truss. In this case the problem is to find the member areas that minimize the weight (volume) while satisfying allowable stress constraints. Here the structural matrices are

\[
N = \begin{bmatrix}
\frac{1}{\sqrt{2}} & 1/\sqrt{2} \\
0 & 1 \\
-1/\sqrt{2} & 1/\sqrt{2}
\end{bmatrix}, \quad K = \begin{bmatrix}
K_1 & 0 & 0 \\
0 & K_2 & 0 \\
0 & 0 & K_3
\end{bmatrix}, \quad P = \begin{bmatrix}
P_x \\
P_y
\end{bmatrix}
\]

The elastic stiffness matrix, \( K_e = N^T K N \), is then
\[
K_E = \begin{bmatrix}
\frac{(K_1 + K_3)}{2} & \frac{(K_1 - K_3)}{2} \\
\frac{(K_1 - K_3)}{2} & \frac{(K_1 + K_3)}{2 + K_2}
\end{bmatrix}
\]

and the inverse stiffness matrix is

\[
K_E^{-1} = \frac{2}{2K_1K_3 + K_2(K_1 + K_3)} \begin{bmatrix}
\frac{(K_1 + K_3)}{2 + K_2} & -\frac{(K_1 - K_3)}{2} \\
-\frac{(K_1 - K_3)}{2} & \frac{(K_1 + K_3)}{2}
\end{bmatrix}
\]

Here \(K_i = A_i E/L_i\), with

- \(A_i\) – area of bar \(i\)
- \(E\) – Young’s modulus
- \(L_i\) – the length of bar \(i\)

The member force matrix \(F\) can now be written in terms of the member stiffnesses as

\[
F = \begin{bmatrix}
F_1 \\
F_2 \\
F_3
\end{bmatrix} = \begin{bmatrix}
\frac{K_1}{\sqrt{2}} \left[ 2(K_1 + K_3)P_x + 2K_1P_y \right] \\
K_2 \left[ -(K_1 - K_3)P_x + (K_1 + K_3)P_y \right] \\
\frac{K_3}{\sqrt{2}} \left[ -2(K_1 + K_2)P_x + 2K_1P_y \right]
\end{bmatrix} \frac{1}{2K_1K_3 + K_2(K_1 + K_3)}
\]

Fig. 1.10. Three-bar Truss

where \(F_i\) is the force in bar \(i\). What emerges is the fact that while the stiffness matrix \(K_E\) is linear in the member stiffnesses (and therefore in the member areas), the
displacements and the member forces are controlled by the inverse of $K_e$, which is a rational function of the member stiffnesses and nontrivial to deal with.

A weight-minimization problem for this truss might be written as follows: Given load matrix $P$ and the structural geometry, find the member areas $A_i$ that minimize the weight while satisfying allowable stress requirements. Formally, find $A_1, A_2, A_3$ which

Minimize $f = A_1L_1 + A_2L_2 + A_3L_3$

Subject to $g_1 = |F_1| - \sigma_1(A_1) A_1 \leq 0$

$g_2 = |F_2| - \sigma_2(A_2) A_2 \leq 0$

$g_3 = |F_3| - \sigma_3(A_3) A_3 \leq 0$

Here the $\sigma_i$ are the allowable stresses which typically vary with the member size and length and the design code used. In the space of areas, surfaces of constant weight (volume) are planes (see Fig. 1.11).

![Fig. 1.11. Area Space](image)

The truss problem just described turns out to be nontrivial to solve as a mathematical programming problem. On the other hand, when optimality criteria methods are discussed later, a version of this problem will be solved using a very simple algorithm. There are two points to be made here. First, there is an interesting tension between classical optimization and optimality criteria methods, which will be considered later. Second, structural optimization is computationally intensive and it is only recently that the tools have become generally available to solve practical structural optimization problems routinely. Appendix B gives a more careful discussion of the truss problem.