Design Optimization using MATLAB® and SOLIDWORKS®

Krishnan Suresh

Design Optimization using MATLAB® and SOLIDWORKS®

A unique text integrating numerics, mathematics and applications to provide a hands-on approach to learning optimization techniques, this mathematically accessible textbook emphasizes conceptual understanding and importance of theorems rather than elaborate proofs. It allows students to develop fundamental optimization methods before delving into MATLAB®'s optimization toolbox, and to link MATLAB's results with the results from their own code. Following a practical approach, the text demonstrates several applications, from error-free analytic examples to truss (size) optimization, and 2D and 3D shape optimization, where numerical errors are inevitable. The principle of minimum potential energy is discussed to highlight the deep relationship between engineering and optimization. MATLAB code in every chapter illustrates key concepts and the text demonstrates the coupling between MATLAB and SOLIDWORKS® for design optimization. A wide variety of optimization problems are covered including constrained non-linear, linear-programming, least-squares, multi-objective, and global optimization problems.

Krishnan Suresh is the Philip and Jean Myers Professor of Mechanical Engineering at the University of Wisconsin–Madison, and a Fellow of the American Society of Mechanical Engineers. He is also the co-founder of SciArt Software (www.sciartsoft.com), a UW–Madison spinoff that creates and supports high-performance topology optimization software solutions.

Design Optimization using MATLAB® and SOLIDWORKS®

Krishnan Suresh
University of Wisconsin–Madison

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India

79 Anson Road, #06–04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/suresh
DOI: 10.1017/9781108869027

First published 2021

Printed in the United Kingdom by TJ Books Limited, Padstow Cornwall

A catalogue record for this publication is available from the British Library.

Library of Congress Cataloging-in-Publication Data
Names: Suresh, Krishnan, 1970– author.
Title: Design optimization using MATLAB and SOLIDWORKS / Krishnan Suresh.
Description: Cambridge, United Kingdom ; New York, NY : Cambridge University Press, 2021. | Series: Cambridge texts in biomedical engineering | Includes bibliographical references and index.
Identifiers: LCCN 2020052134 | ISBN 9781108491600 (hardback) | ISBN 9781108869027 (ebook)
Subjects: LCSH: MATLAB. | SolidWorks. | Engineering design – Data processing – Textbooks. | Mathematical optimization – Textbooks.
Classification: LCC TA174 .S925 2021 | DDC 620/.00420285536–dc23
LC record available at https://lccn.loc.gov/2020052134

ISBN 978-1-108-49160-0 Hardback

Additional resources for this publication at www.cambridge.org/highereducation/suresh

Dedicated to my family.

They mean the world to me.

Contents

Preface *page* xiii

1 Introduction 1

1.1 An Example of Optimization 1
1.2 Challenges 1
1.3 MATLAB Code 3
1.4 Organization of Text 3

2 Modeling 7

2.1 Standard Optimization Formulation 7
2.2 Illustrative Examples 9
2.3 Geometry Problems 10
2.4 Analytical Design Problems 16
2.5 Structural Analysis 18
2.6 Structural Optimization 23
2.7 Conclusions and Observations 25
2.8 Exercises 26

3 Introduction to MATLAB 30

3.1 Basics 30
3.2 MATLAB Code Resource 33
3.3 Memory Pre-allocation and Vectorization 34
3.4 MATLAB Script Files 34
3.5 Linear Algebra 36
3.6 Complex Numbers 37
3.7 Plots 37
3.8 Symbolic Operations 39
3.9 User-Defined Functions 40
3.10 Variable Arguments 41
3.11 Solving Non-linear Equations 42
3.12 Modules 44
3.13 Sampling Algorithm 45
3.14 Polynomial Class 47
3.15 Extending the Polynomial Class 51
3.16 Exercises 52

4 Unconstrained Optimization: Theory 59

4.1 Local versus Global Minimum 60
4.2 First-Order Condition 61
4.3 Second-Order Condition 65
4.4 Sylvester's Criterion 68
4.5 Quadratic and Convex Functions 69
4.6 Illustrative Examples 70
4.7 Exercises 76

5 Unconstrained Optimization: Algorithms 79

5.1 Test Functions 79
5.2 One-Variable Minimization 81
5.3 Unimodal Functions 82
5.4 Termination Criteria 83
5.5 1D Methods 84
5.5.1 Trisection 84
5.5.2 Golden Section 87
5.5.3 Quadratic Interpolation 89
5.5.4 Bisection 90
5.5.5 Newton–Raphson 90
5.6 Multi-variable Minimization 95
5.7 Line-Search 97
5.8 Search Directions 99
5.9 Eigendirections 101
5.10 Conjugate Directions 102
5.11 Powell's Method 103
5.12 Gradient Methods 106
5.12.1 Steepest-Descent Method 106
5.12.2 Linear Conjugate Gradient Method 109
5.12.3 Non-linear Conjugate Gradient Method 111
5.13 Newton–Raphson Method 112
5.14 Equilibrium of Spring Systems 113
5.15 Summary 120
5.16 Exercises 121

6 MATLAB Optimization Toolbox 126

6.1 Overview 126
6.2 The `fminbnd` Function 127
6.3 The `fzero` Function 132
6.4 The `fminsearch` Function 134
6.5 The `fminunc` Function 137

6.6 The `fsolve` Function 142
6.7 Global Optimization Toolbox 144
6.7.1 `MultiStart` 145
6.7.2 `GlobalSearch` 146
6.7.3 Genetic Algorithms 147
6.7.4 Simulated Annealing 147
6.7.5 Multiple Local Minima 148
6.7.6 Noisy Objective 150
6.8 Exercises 151

7 Constrained Optimization 155

7.1 Equality Constraints 155
7.2 Optimality Criteria: Equality Constraints 158
7.2.1 Single Equality Constraint 158
7.2.2 Quadratic Problems 160
7.2.3 Multiple Constraints 161
7.2.4 Significance of Lagrange Multipliers 162
7.3 Optimality Criteria: Inequality Constraints 164
7.4 The `fmincon` Method 171
7.4.1 Basics 172
7.4.2 Exploiting Gradients 176
7.5 Constrained Spring Problem 178
7.6 The `fminsearchcon` Function 182
7.6.1 Numerical Challenges 183
7.7 Conclusions 186
7.8 Exercises 186

8 Special Classes of Problems 195

8.1 Linear Programming 196
8.2 Mixed Integer Linear Programming 198
8.3 Least-squares Problems 200
8.3.1 Linear Least Squares 200
8.3.2 Non-linear Least Squares 203
8.4 Multi-objective Optimization 204
8.5 Exercises 206

9 Truss Analysis 209

9.1 Overview 209
9.2 Conventions 210
9.3 Small-Displacement Assumption 211

9.4 Force Balance Method 212
9.4.1 Determinate Truss System 213
9.4.2 Indeterminate Truss System 215
9.5 Potential Energy Method 218
9.6 Assembly of Truss Linear System 219
9.7 Truss Modeling Using MATLAB 220
9.8 Exercises 229

10 Size Optimization of Trusses 235

10.1 Compliance Minimization 235
10.1.1 Determinate Truss System 237
10.2 Compliance Minimization Using MATLAB 240
10.2.1 Direct Implementation 241
10.2.2 Scaling the Design Variables 244
10.2.3 Scaling the Constraints and Objective 245
10.2.4 Indeterminate Truss System 248
10.3 Compliance-Constrained Volume Minimization 250
10.4 Stress-Constrained Volume Minimization 253
10.4.1 Stress Constraints: Determinate Truss 253
10.4.2 Stress Constraints: MATLAB Implementation 256
10.5 Buckling Constraints 259
10.5.1 Buckling Constraints: Determinate Truss 260
10.6 Exercises 261

11 Gradient Computation 267

11.1 Finite Difference in 1D 267
11.2 Finite Difference in N Dimensions 272
11.3 Analytical Approach 276
11.4 Complex-Variable Approach 278
11.5 Gradient of Compliance 282
11.6 Automatic Differentiation 284
11.7 Exercises 285

12 Finite Element Analysis in 2D 287

12.1 Overview 287
12.2 Analysis of a Vertical Bar 288
12.2.1 Tensile Load 290
12.2.2 Bending Load 292
12.2.3 Bending Load with Finer Mesh 294

12.3 MATLAB Implementation 295
12.3.1 `Brep2D` 295
12.3.2 `TriMesher` 295
12.3.3 `TriElasticity` 296
12.4 Analysis of Cantilever Beam 296
12.5 Analysis of L-bracket 299
12.6 Analysis of a Plate 301
12.7 Exercises 304

13 Shape Optimization in 2D 312

13.1 Shape Optimization and Shape Parameters 312
13.2 Parametric Studies 313
13.3 Compliance and Volume Minimization 316
13.4 Scaling for Numerical Robustness 317
13.5 Numerical Noise from FEA 318
13.6 Finite-Difference Step Size 320
13.7 Semi-analytic Gradients in FEA 323
13.8 Global Search Method 326
13.9 Stress Scatter Plot 327
13.10 Geometric Constraints 327
13.11 Conclusions 328
13.12 Exercises 328

14 Finite Element Analysis in 3D 332

14.1 Overview 332
14.2 Analysis of Cantilever Beam 332
14.3 Analysis of L-bracket 334
14.4 Analysis of Knuckle 336
14.5 Exercises 339

15 SOLIDLAB: A SOLIDWORKS–MATLAB Interface 343

15.1 Overview and Installation 343
15.2 Geometric Queries 345
15.3 FEA Queries 349
15.4 Displacement Scatter Plot 351
15.5 Stress Scatter Plot 352
15.6 Conclusions 354
15.7 Exercises 354

16 Shape Optimization Using SOLIDLAB 357

16.1 Overview 357
16.2 Displacement Minimization 357
16.3 Displacement Minimization Using Global Search 361
16.4 Stress Minimization 363
16.5 Conclusions 366
16.6 Exercises 366

Appendix 369

A.1 Taylor Series 369
A.2 Optimality Theorems 371

References 375
Index 377

Preface

Origins of This Text

As is of true of many textbooks, this text has evolved from teaching a formal course, specifically, "Optimum Design of Mechanical Elements and Systems," at the University of Wisconsin–Madison. It has undergone several revisions over the last decade.

Target Audience

The primary audience for this text includes senior undergraduate students, junior graduate students and practicing engineers. Given this wide audience, prerequisites are kept to a minimum. Basic undergraduate-level mathematics is assumed, but no prior background in optimization is required. Prior experience in programming, especially MATLAB, is helpful.

Topics Covered

This text covers three complementary topics in optimization: (1) the underlying mathematics, (2) numerical methods and nuances and (3) engineering applications.

On the first topic, we will recall basic results from single- and multi-variable calculus. Emphasis is given to conceptual understanding of theorems rather than to elaborate proofs (which can be found in mathematically oriented texts).

The second topic, numerical methods, lies at the heart of optimization. It is impossible to understand optimization without a good grasp of numerical methods. Therefore, this text encourages the reader to implement simple optimization methods "from scratch," before delving into MATLAB's optimization toolbox. This will provide a good understanding of how optimization algorithms work, or sometimes fail! Besides the generic class of constrained non-linear optimization problems, we will also discuss special types of problems, including linear-programming problems, least-squares problems and multi-objective problems, that require special treatment. Finally, we will briefly explore global optimization methods that are becoming increasingly important in engineering.

It is the author's strong opinion that engineers should learn optimization within the context of applications. This text largely focuses on geometric and structural applications. For example, we will study how truss systems can be optimized, and some of the pitfalls. Lessons learned from this exercise will be applied toward the structural shape optimization of 2D and 3D designs. The reader is encouraged to apply underlying optimization principles and methods to his/her field of study.

Topics Not Covered

Engineering optimization is too broad to be covered in a single text. Consequently, a few topics have been omitted; these include surrogate modeling, stochastic optimization and optimization under uncertainty.

Software Resources

This text assumes that the reader has access to MATLAB® (www.mathworks.com). The MATLAB code accompanying this text is an integral part of student learning, and can be downloaded from the author's website at www.ersl.wisc.edu under the "Research" tab. The MATLAB code is object-oriented; it teaches the basic concepts of data encapsulation, inheritance and code reuse. Through exercises, the reader will learn to extend the code to address various optimization problems.

In the last couple of chapters, we will create and analyze 3D designs using SOLIDWORKS® (www.solidworks.com). It is assumed that the reader is familiar with creating simple models in SOLIDWORKS. Basic finite element analysis using SOLIDWORKS is covered in the text. Then a toolbox, namely SOLIDLAB, which serves as an interface between SOLIDWORKS and MATLAB, is presented; SOLIDLAB was developed by the author's research group. Through SOLIDLAB, one can, for example, modify feature dimensions of SOLIDWORKS models, query mass properties, carry out a finite element analysis and optimize, all from within the comfort of MATLAB.

Acknowledgments

Textbook writing inevitably takes time away from family, and I would like to acknowledge the support of my dear wife, Vanitha, for her constant encouragement to finish this text. Meanwhile, our two sons, Sanjay and Arjun, have smartly learned to work around my busy schedule. I would also like to thank my graduate students:

Aaditya Chandrasekhar, Josh Danczyk, Tej Kumar, Amir M. Mirzendehdel, Bhagyashree Prabhune, Saketh Sridhara and Subodh Subedi, who have contributed both directly and indirectly to this text. I would also like to acknowledge the support of the National Science Foundation and the Graduate School at the University of Wisconsin–Madison; they have helped nurture this text.

1 Introduction

Optimization is an integral part of engineering today. Engineers use optimization techniques to design civil structures, machine components, electrical circuits, plant layouts, chemical processes and so on. Indeed, one simply cannot make technological advances without optimization.

1.1 An Example of Optimization

While there are numerous examples of optimization, we will consider here a design problem posed in Figure 1.1 where a plate is subject to a uniform pressure loading on one face, and is fixed at the other. The plate can be modeled and analyzed using any of the popular finite element packages; we will discuss one such package, namely SOLIDWORKS ([1]), later in the text.

Based on finite element analysis (FEA), one can determine the stress distribution within the plate, as illustrated in Figure 1.2; the maximum stress happens to be around 515 MPa, and occurs on the periphery of the large hole, as expected.

A typical design objective now is to reduce the maximum stress, without increasing the mass of the plate. Further, the overall plate dimensions and the diameter of the larger hole cannot be modified. The location and size of the two smaller holes can, however, be modified. In this example, the engineer finds out, through trial-and-error, that the stress can be reduced by simply enlarging the two smaller holes. This is illustrated in Figure 1.3, where the maximum stress is reduced to 466 MPa; this is done by increasing the diameter of the two holes from 15 mm to 25 mm. The maximum stress now occurs at the periphery of one of the smaller holes. *Observe that, in the process of reducing the stress, the mass has also been reduced!*

One can now ask if the stress can be further reduced, i.e., *is there an optimal location and optimal diameter for the two smaller holes such that the maximum stress is minimized?* This is an example of shape optimization that we shall study later in the text.

1.2 Challenges

There are numerous such "simple" optimization examples in engineering. However, in the author's experience, while solving such problems engineers often run into several challenges: (1) *How does one translate the above problem*

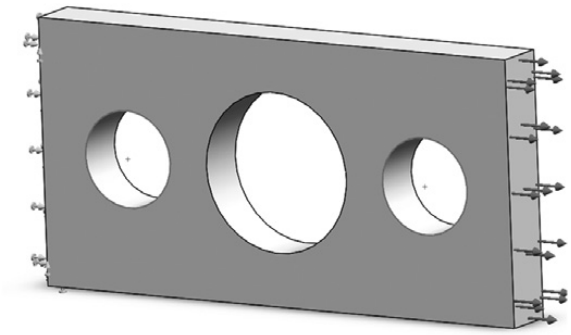

Figure 1.1 Design problem.

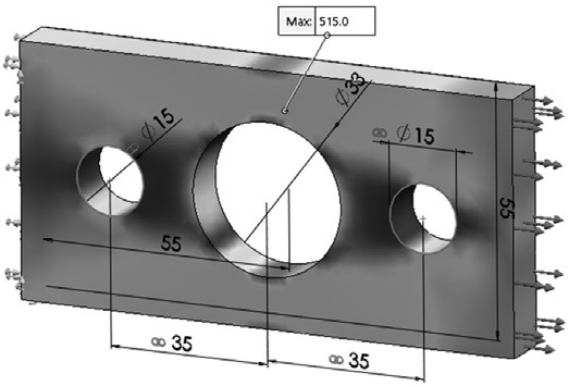

Figure 1.2 Stress plot based on finite element analysis (FEA).

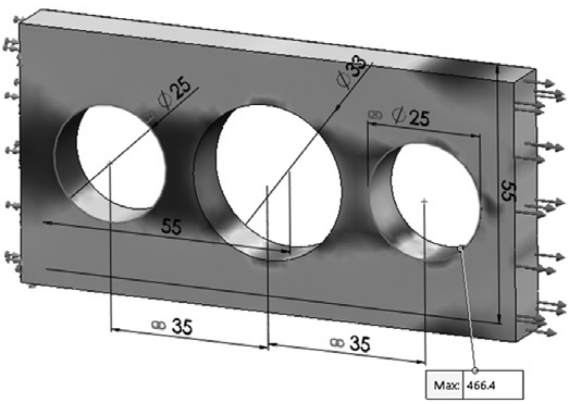

Figure 1.3 Reduced stress for a modified design.

into a formal optimization statement (with objective, constraints, feasible space, design variables, etc.)? (2) *What optimization method should one use and why?* (3) *What if the optimization method does not converge?* And so on.

After reading through this text, and completing the exercises, the author sincerely believes that the reader will be able to answer such questions confidently, and also extend the concepts to his/her field of study.

Study of optimization can be fun and enriching. However, for most effective learning, engineers should study optimization within the context of a relevant and familiar application. Learning to code sophisticated optimization methods, without understanding why these different methods exist in the first place, is both meaningless and counterproductive. Therefore, this text introduces fundamental optimization concepts through concrete applications.

For example, to introduce the important concept of *numerical scaling*, we will consider optimizing a truss system. We will observe that, without numerical scaling, even the best optimization method will not converge correctly. By considering a variety of such applications, fundamental optimization concepts can be assimilated easily.

No prior background in optimization is assumed in this text; basic undergraduate-level mathematics is sufficient. Prior experience in programming, especially MATLAB programming [2], is helpful. For this text, we will rely entirely on MATLAB programming, basics of which are covered in Chapters 3, 6 and 7.

1.3 MATLAB Code

The MATLAB code accompanying this text is an integral part of student learning, and can be downloaded as a zip-file from the author's website at www.ersl.wisc.edu under the "Research" tab. The code is organized chapter-wise; the use of this code is discussed in Chapter 3, where MATLAB is introduced.

1.4 Organization of Text

This text covers three complementary topics in engineering optimization: (1) the underlying mathematics, (2) numerical methods and nuances and (3) engineering applications; see Figure 1.4.

The text is organized as in Table 1.1; each chapter will introduce a critical mathematical concept/numerical method (identified by rows in the table) by considering a specific engineering application (identified by columns in the table).

Chapter 2 introduces the critical concept of *modeling*, i.e., the art of translating a loosely worded optimization problem into a formal mathematical statement. The notions of objective and constraint are introduced by considering various applications. This chapter will serve as an overview for the remainder of this text.

Chapter 3 is a short introduction to MATLAB. It is by no means an exhaustive review. The reader will be introduced to basic computing and plotting

Table 1.1 Text organization across various chapters.

Applications → *Concepts/Tools*↓	Analytical optimization	Truss optimization	Shape optimization
Modeling	Chapter 2	Chapter 2	Chapter 2
Basic MATLAB programming	Chapter 3		
Unconstrained theory	Chapter 4		
Basic algorithms	Chapter 5		
MATLAB optimization toolbox	Chapter 6		
Constrained theory	Chapter 7		
Specialized problems	Chapter 8		
Structural analysis		Chapter 9	
Numerical scaling		Chapter 10	
Gradient computation		Chapter 11	
Finite element analysis (2D)			Chapters 12, 13
Finite element analysis (3D)			Chapters 14, 15, 16

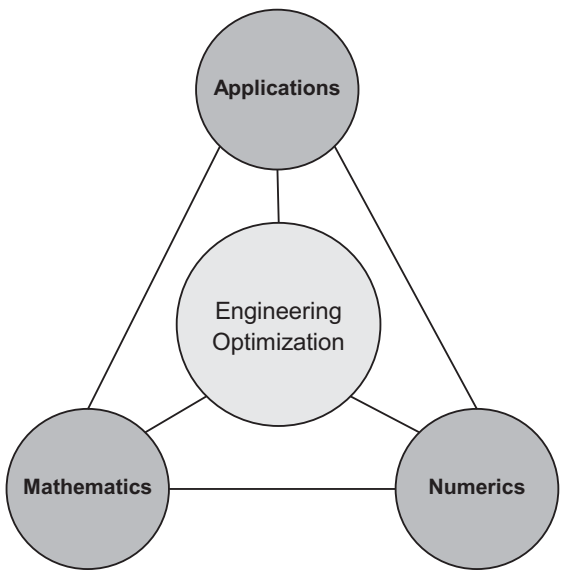

Figure 1.4 Engineering optimization encompasses three topics: mathematics, numerics and applications.

routines available within MATLAB. Various programming constructs such as the for-loop and if-then-else will be discussed. The concept of object-oriented programming will be reviewed through examples.

In Chapter 4, we will address optimization theory, focusing on the *unconstrained optimization* problems discussed in Chapter 2. Some of the questions raised in Chapter 2 will be answered. Critical concepts such as global/local minima and stationary points will be introduced.

Chapter 5 complements Chapter 4 in that we will *implement a few basic algorithms* to appreciate the nuances of numerical optimization. While engineers do not typically implement optimization algorithms "from scratch," numerical implementation will provide a good understanding of how algorithms work, and sometimes fail!

In Chapter 6, we will delve into MATLAB's *optimization toolbox* and study some of the algorithms for solving unconstrained problems. Using several test cases, we will observe the behavior of these methods, and correlate them to the observations made in the previous chapter.

Chapter 7 addresses the basic theory behind *constrained optimization*. Critical concepts such as Lagrange multipliers will be introduced within the context of engineering applications. Physical and mathematical interpretations of Lagrange multipliers will be discussed. We will also study MATLAB-supported algorithms for solving constrained problems.

Chapter 8 covers certain special types of optimization problems, including linear-programming problems, least-squares problems and multi-objective problems, that are not addressed in the previous chapters. One of the primary reasons for treating them separately is that they often require the use of specialized numerical methods.

In Chapter 9, trusses are analyzed through force balance and potential energy principles. The main concept emphasized is that "elastic structures, such as truss systems, when subject to an external load, reach a stable configuration when their potential energy reaches a local minimum." In other words, *physical systems behave in an optimal fashion*! This observation is both fascinating and important in engineering.

Chapter 10 will build on Chapter 9 by considering the *size optimization of truss systems*, where the goal is to find the optimal size (diameter) of truss members such that a given objective (say, the mass of the truss) is minimized, subject to certain constraints (such as deflection and stresses). MATLAB optimization methods introduced in Chapter 7 will be deployed to find optimal solutions to such problems. We will find that the algorithms do not always converge to the correct answer (even for simple problems). This will motivate the need for *numerical scaling* – a critical concept in numerical optimization! This concept will be covered and illustrated through several examples.

Chapter 11 will cover an equally important concept of gradient (i.e., sensitivity) computation. Gradient computation is critical if first-order methods of optimization are employed. This chapter covers the pitfalls of finite-difference-based gradient computation and provides alternative methods.

In Chapters 12 and 13, we consider FEA and optimization of 2D elastic problems. Conceptually, this is a direct extension of truss analysis and

optimization, discussed in Chapters 9 and 10 respectively. However, new concepts such as shape parameters and finite element discretization enter the picture.

Chapter 14 is an extension of Chapter 12 to 3D FEA. Here, we will rely on SOLIDWORKS for modeling and analysis. It is assumed that the reader is familiar with the process of creating 3D models and carrying out basic FEA within SOLIDWORKS. The objective of this chapter is to develop a foundation for parametric study and optimization to be pursued in the next chapter.

Chapter 15 introduces SOLIDLAB, an interface between SOLIDWORKS and MATLAB. SOLIDLAB was developed by the author and his graduate students. This chapter will illustrate the use of SOLIDLAB to query and analyze SOLIDWORKS models, from within the comfort of MATLAB.

Chapter 16 addresses 3D shape optimization by combining SOLIDWORKS, MATLAB and SOLIDLAB. The fundamental difference between compliance and stress minimization is highlighted.

The Appendix covers additional mathematical concepts and proofs.

Finally, it goes without saying that no textbook is ever complete or comprehensive. There are several excellent textbooks on engineering optimization (see references [3], [4], [5], [6], [7]) and numerical optimization (see references [8], [9]) that complement this text. The reader is strongly encouraged to consult these for topics not covered here.

2 Modeling

Highlights

1. This chapter discusses *modeling*, the first step in optimization. Modeling is the process of converting a loosely worded optimization problem into a formal mathematical statement.
2. Several modeling examples from geometry and structural mechanics are considered, and each example is converted into the standard formulation.
3. Through these examples, relevant optimization terminology is also explained.
4. Finally, important observations are made about modeling; these include: (1) introducing appropriate design variables, (2) exploiting optimality criteria for simplification and (3) the iterative nature of modeling.

Modeling is the process of converting a loosely worded "optimization" problem into a mathematically precise and standard formulation. For example, converting the stress minimization problem discussed in Section 1.1 into a formal optimization statement would be a modeling effort. Accurate modeling is crucial; an inaccurate model will lead to erroneous conclusions. To illustrate the concept of modeling, we consider several examples in this chapter. However, first, the standard optimization formulation is presented, together with an explanation of the terminology.

2.1 Standard Optimization Formulation

Almost all optimization problems considered in this text will be posed in the following standard form ("s.t." is short for "such that"):

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h_i(\mathbf{x}) = 0;\ i = 1, 2, \ldots \\ & g_j(\mathbf{x}) \leq 0;\ j = 1, 2, \ldots \\ & \mathbf{x}^{\min} \leq \mathbf{x} \leq \mathbf{x}^{\max} \\ & \mathbf{x} = \{x_1 \quad x_2 \quad \ldots \quad x_N\} \end{aligned} \tag{2.1}$$

where

- $f(\mathbf{x})$ is the single *objective* that we are trying to minimize. Given a loosely worded problem statement, the first task is to identify and *quantify* the objective. In structural design problems, the objective is typically the volume or weight of a part, or the maximum deflection, or the maximum stress, and so on. Two points are worth noting here: (1) Often, an engineer may be interested in multiple objectives (for example, minimize weight and minimize stress); such *multi-objective* problems are briefly considered in Section 6.7, but are not the main focus of this text; further, it is often possible to interpret one or more of these objectives as constraints. (2) If we desire to *maximize* an objective (for example, maximize the stiffness of a part), it is easy to convert this into a minimization problem in multiple ways, as discussed later on.
- $\mathbf{x}$ are the *optimization or design variables*. These are the free parameters that can be modified to meet the objective. In structural design problems, these could be geometric parameters (thickness of a truss member, the location and size of a hole, topology), or material properties (Young's modulus, yield strength), and so on. In this text, we will assume that the optimization variables are *continuously varying* (such as the radius of a hole). Discrete variables, such as the number of holes in a design, are not explicitly treated in this text; however, references and examples are provided on how such integer problems can be handled.
- $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$ are the lower and upper bounds on the optimization variables; for example, a lower bound and/or upper bound on a hole radius. It is not essential for optimization variables to exhibit lower and upper bounds. However, if such bounds exist, it is important to include them in the problem statement.
- $h_i(\mathbf{x})$ are the *equality constraints*; for example, "the diameter of truss member A must be exactly equal to three times the diameter of truss member B." While we are not distinguishing here between linear and non-linear constraints in the formulation, such differences can be important in numerical analysis. They are highlighted later on.
- $g_j(\mathbf{x})$ are the *inequality constraints*; for example, "the diameter of truss member A must be less than three times the diameter of truss member B." Again, we are not distinguishing here between linear and non-linear inequality constraints, but such differences can be important in numerical analysis.

In the remainder of this chapter, we shall study several optimization examples and convert them into the standard form shown in Equation (2.1). We will observe that there are several special cases of the standard form; for example, when the constraints are absent, we obtain an *unconstrained minimization* problem, which is mathematically and numerically easy to analyze:

$$\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x}) \tag{2.2}$$

where $\mathbf{x} = \{x_1 \quad x_2 \quad \ldots \quad x_N\}$. Further, when there is precisely one optimization variable, this reduces to a *single-variable unconstrained minimization*, which is one of the simplest optimization problems that one can pose:

$$\underset{x}{\text{minimize}} \; f(x) \tag{2.3}$$

2.2 Illustrative Examples

Before moving to modeling, we consider here several examples of the standard form. Consider

$$\underset{x}{\text{minimize}} \; x^2 \tag{2.4}$$

This is a single-variable unconstrained minimization problem, with a trivial solution of $x = 0$. On the other hand, the problem

$$\underset{x}{\text{minimize}} \; 3 \sin x - x + 0.1x^2 + 0.1 \cos(2x) \tag{2.5}$$

is also a single-variable unconstrained minimization problem, but its solution will require numerical analysis. The problem

$$\underset{\{u,v\}}{\text{minimize}} \begin{Bmatrix} \frac{1}{2}100\left(\sqrt{u^2 + (1+v)^2} - 1\right)^2 + \frac{1}{2}50\left(\sqrt{u^2 + (1-v)^2} - 1\right)^2 \\ -(10u + 8v) \end{Bmatrix} \tag{2.6}$$

is a two-variable unconstrained minimization problem. The physical interpretation of the above problem is discussed later in this chapter. One can add upper and lower bounds to the above problem, leading to

$$\begin{aligned} \underset{\{u,v\}}{\text{minimize}} \; & \begin{Bmatrix} \frac{1}{2}100\left(\sqrt{u^2 + (1+v)^2} - 1\right)^2 + \frac{1}{2}50\left(\sqrt{u^2 + (1-v)^2} - 1\right)^2 \\ -(10u + 8v) \end{Bmatrix} \\ \text{s.t.} \quad & 0 \le u \le 1.0 \\ & 0 \le v \end{aligned} \tag{2.7}$$

One can also impose an equality constraint linking the two variables:

$$\begin{aligned} \underset{\{u,v\}}{\text{minimize}} \; & \begin{Bmatrix} \frac{1}{2}100\left(\sqrt{u^2 + (1+v)^2} - 1\right)^2 + \frac{1}{2}50\left(\sqrt{u^2 + (1-v)^2} - 1\right)^2 \\ -(10u + 8v) \end{Bmatrix} \\ \text{s.t.} \quad & v - u^3 = 0 \end{aligned} \tag{2.8}$$

The physical meaning behind such constraints is discussed later in this chapter, and such constraints can completely change the complexity of the problem.

2.3 Geometry Problems

We now consider problems in geometry; the primary goal is to translate each of these problems into the standard form presented in Equation (2.1).

Example 2.1 Closest-Point Computation

Problem: Given a point p on a plane, and a straight line passing through the origin, find the closest point on the straight line to the given point p.

Modeling: Let the straight line passing through the origin be denoted by $y = mx$ and the given point be $p = (x_0, y_0)$. The task is to find the point q on the straight line that is closest to p; see Figure 2.1.
Let $q = (\alpha, m\alpha)$ be the point on the straight line. Observe that q is defined such that it always lies on the straight line $y = mx$. Introducing such intermediate variables that satisfy the problem constraints is one of the most important steps in modeling. Observe that the only unknown now is α.

Finding a point q closest to p is equivalent to minimizing the distance between them, where the distance is given by

$$D(\alpha) = \sqrt{(x_0 - \alpha)^2 + (y_0 - m\alpha)^2} \tag{2.9}$$

Thus, the optimization problem is posed as

$$\underset{\alpha}{\text{minimize}} \; D(\alpha) = \sqrt{(x_0 - \alpha)^2 + (y_0 - m\alpha)^2} \tag{2.10}$$

Equation (2.10) is interpreted as "Find α that minimizes $D(\alpha)$."

Classification: The problem posed in Equation (2.10) is *a single-objective, single-variable, unconstrained* optimization problem since: (1) there is only one objective namely, the distance D, that must be minimized, (2) there is only one continuous (unknown) variable α and (3) there are no constraints.

Solution: In the chapters that follow, we shall discuss various numerical methods to solve optimization problems. However, for now, recall from basic calculus that, if a continuous *differentiable* function takes a minimum at a point, then the derivative of

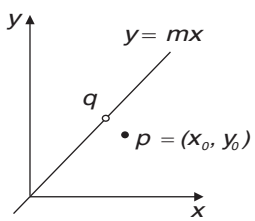

Figure 2.1 Closest point on a straight line.

(cont.)

that function vanishes at that point. Thus, differentiating the objective $D(\alpha)$ with respect to α and setting it equal to zero will yield the value of α:

$$\frac{dD}{d\alpha} = \frac{-2(x_0 - \alpha) - 2m(y_0 - m\alpha)}{2\sqrt{(x_0 - \alpha)^2 + (y_0 - m\alpha)^2}} = 0 \tag{2.11}$$

i.e.,

$$\alpha = \frac{x_0 + my_0}{(1 + m^2)} \tag{2.12}$$

Thus, the closest point is

$$q = \left(\frac{x_0 + my_0}{(1 + m^2)}, \frac{mx_0 + m^2 y_0}{(1 + m^2)}\right) \tag{2.13}$$

and the shortest distance is

$$D = \frac{|(x_0 m - y_0)|}{\sqrt{(1 + m^2)}} \tag{2.14}$$

Caveat: We have not formally established that Equation (2.11) yields a minimum; we shall address this in later chapters.

Verification: Verify whether the above solution yields the "expected answer." For example, suppose the straight line is the x axis, and $p = (0, 1)$: do you recover the expected solution? Verification, if possible, is highly recommended in order to catch modeling errors.

Example 2.2 Fitting a Straight Line

Problem: Yet another commonly occurring optimization problem is "data-fitting," where, given a set of points, the task is to find a straight line that best fits the set of points.

Modeling: Let the set of 2D data points be (x_i, y_i), $i = 1, 2, \ldots, N$; one must find the straight line that best fits the data-set (see Figure 2.2).

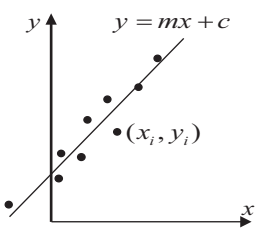

Figure 2.2 Find the best-fitting straight line.

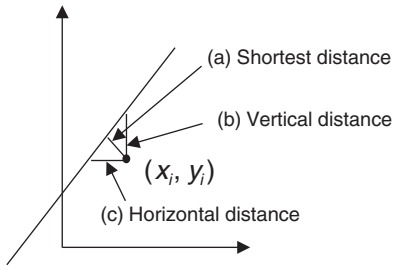

Figure 2.3 Various interpretations of deviation.

The overall strategy is to construct a straight line that minimizes the total deviation from the data points. There are, however, various definitions for "deviation" (see Figure 2.3): (a) the *shortest distance* between the data point and straight line, (b) the *vertical distance* between the data point and straight line and (c) the *horizontal distance* between the data point and straight line. Each of these will yield a different mathematical problem and solution!

This highlights a second important aspect of modeling: *a problem description can often be interpreted in multiple ways, yielding different solutions*. Here, we choose to minimize the vertical distance.

Let the straight line be of the form $y = mx + c$. The task then is to find the optimal values of m and c. Since deviation is defined as the vertical distance between a data point and the straight line, for a given point (x_i, y_i) the deviation (i.e., error) is given by

$$E_i = |mx_i + c - y_i| \tag{2.15}$$

The "best-fitting" straight line can be defined as the one that minimizes the sum of squares of all errors, i.e.,

$$\underset{m,c}{\text{minimize}} \; E = \sum_{i=1,2,\ldots}^{N} (mx_i + c - y_i)^2 \tag{2.16}$$

Equation (2.16) is interpreted as "find m and c that minimize the sum of squares of individual deviations." The reason for squaring the error measure is to make the objective differentiable, an important consideration in optimization.

Classification: The problem posed in Equation (2.16) is a *single-objective, two-variable, unconstrained* optimization problem since: (1) there is only one objective, namely the total error E, that needs to be minimized, (2) there are two continuous (unknown) variables, m and c, and (3) there are no constraints.

Solution: For multi-variable unconstrained problems, we set the partial derivative of the objective with respect to each variable to zero (this is discussed formally in later chapters), i.e.,

$$\frac{\partial E}{\partial m} = 0 \Rightarrow \sum_{i=1,2,\ldots}^{N} 2(mx_i + c - y_i)x_i = 0 \tag{2.17}$$

and

$$\frac{\partial E}{\partial c} = 0 \Rightarrow \sum_{i=1,2,\ldots}^{N} 2(mx_i + c - y_i) = 0 \tag{2.18}$$

Thus we have two linear equations involving two unknowns:

$$m \sum_{i=1,2,\ldots}^{N} (x_i)^2 + c \sum_{i=1,2,\ldots}^{N} x_i - \sum_{i=1,2,\ldots}^{N} x_i y_i = 0 \tag{2.19}$$

and

$$m \sum_{i=1,2,\ldots}^{N} x_i + c \sum_{i=1,2,\ldots}^{N} 1 - \sum_{i=1,2,\ldots}^{N} y_i = 0 \tag{2.20}$$

One can express the two equations in a standard linear algebraic form:

$$\begin{bmatrix} \sum_{i=1,2,\ldots}^{N} (x_i)^2 & \sum_{i=1,2,\ldots}^{N} x_i \\ \sum_{i=1,2,\ldots}^{N} x_i & \sum_{i=1,2,\ldots}^{N} 1 \end{bmatrix} \begin{Bmatrix} m \\ c \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1,2,\ldots}^{N} x_i y_i \\ \sum_{i=1,2,\ldots}^{N} y_i \end{Bmatrix} \tag{2.21}$$

Many optimization problems reduce to linear algebra problems involving symmetric matrices. Equation (2.21) can be solved provided the 2×2 matrix is invertible.

Verification: As a specific example, consider finding the best-fitting straight line to the two data points (0, 0) and (1, 1). Equation (2.21) reduces to

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} m \\ c \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \tag{2.22}$$

One can verify that the solution is $m = 1$ and $c = 0$ (as expected).

Example 2.3 Area Maximization

Problem: Construct the largest rectangle within a circle of radius R. Figure 2.4 illustrates three sub-optimal rectangles within a circle.

(cont.)

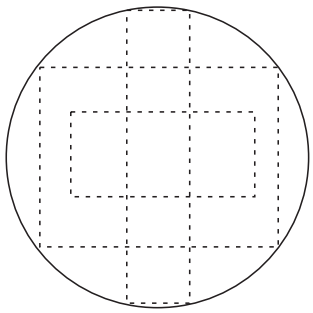

Figure 2.4 Find the largest rectangle within a circle.

Modeling: We leave it as an exercise for the reader to show that, for a rectangle within a circle to have maximum area, all four corners of the rectangle must touch the circle (a necessary condition). This *optimality condition* (a condition that must be satisfied when the solution is optimal) simplifies the problem considerably.

Thus, one of the rectangles in Figure 2.4 is clearly non-optimal. Now let an optimal rectangle (with corners touching the circle) be of width W and height H. Observe that since the corners touch the circle, W and H are not independent. Indeed, we must have the following constraint:

$$\left(\frac{W}{2}\right)^2 + \left(\frac{H}{2}\right)^2 = R^2 \tag{2.23}$$

Further, since the area is given by WH, the optimization problem reads

$$\begin{aligned} &\underset{\{W,H\}}{\text{maximize}} \quad (WH) \\ &\text{s.t.} \qquad \left(\frac{W}{2}\right)^2 + \left(\frac{H}{2}\right)^2 = R^2 \end{aligned} \tag{2.24}$$

Observe that this is a *maximization* problem with one objective, two variables and one equality constraint. One can easily transform this into the standard minimization problem by introducing a negative sign to the objective:

$$\begin{aligned} &\underset{\{W,H\}}{\text{minimize}} \quad (-WH) \\ &\text{s.t.} \qquad \left(\frac{W}{2}\right)^2 + \left(\frac{H}{2}\right)^2 = R^2 \end{aligned} \tag{2.25}$$

Solution: Later in the text, we shall consider formal methods for solving optimization problems with constraints. But, for now, we will introduce an auxiliary variable θ such that the constraint is automatically satisfied and can therefore be eliminated. In particular, let

$$\begin{aligned} W &= 2R\cos\theta \\ H &= 2R\sin\theta \end{aligned} \tag{2.26}$$

Observe that this ensures that the equality constraint is always satisfied. Thus, one can pose Equation (2.24) as

$$\underset{\{\theta\}}{\text{minimize}} \quad -4R^2\sin\theta\cos\theta \tag{2.27}$$

i.e., we have reduced the problem to a one-variable unconstrained problem. As the reader can verify, the solution is $\theta = \pi/4$, i.e.,

$$\begin{aligned} W &= R\sqrt{2} \\ H &= R\sqrt{2} \end{aligned} \tag{2.28}$$

and the maximum area is $2R^2$. Is the answer reasonable?

Example 2.4 Container Optimization

Problem: Consider an empty cylinder of radius R and height H that is closed at both ends. The objective is to minimize the surface area of the cylinder while ensuring that it contains a certain volume. This has practical implications – for example, minimizing the material usage of a soda-can of a given volume.

Modeling: The total surface area of the cylinder is $A = 2\pi R^2 + 2\pi RH$. Further, the volume of the cylinder is $V = \pi R^2 H$. Thus, one can state the optimization problem as follows:

$$\begin{aligned} &\underset{\{R,H\}}{\text{minimize}} \quad (2\pi R^2 + 2\pi RH) \\ &\text{s.t.} \qquad \pi R^2 H - V_0 = 0 \end{aligned} \tag{2.29}$$

Equation (2.29) represents a *minimization problem involving two variables, with an equality constraint*. Observe that one can eliminate the constraint in Equation (2.29) by substituting $H = V_0/(\pi R^2)$ in the objective (we shall discuss pitfalls of constraint elimination later on), simplifying the problem to

$$\underset{\{R\}}{\text{minimize}} \quad \left(2\pi R^2 + \frac{2V_0}{R}\right) \tag{2.30}$$

Observe that we have reduced the problem to a single-variable, unconstrained minimization problem.

Solution: Once again we set the derivative of the objective to zero:

$$\begin{aligned} &\frac{d\left(2\pi R^2 + \frac{2V_0}{R}\right)}{dR} = 0 \;\Rightarrow\; 4\pi R - \frac{2V_0}{R^2} = 0 \\ &\Rightarrow \quad R = \sqrt[3]{\frac{V_0}{2\pi}} \\ &\quad H = V_0/(\pi R^2) \end{aligned} \tag{2.31}$$

(cont.)

Instances: Now, suppose V_0 = 16oz (0.00047 m^3). We have

$$R = \sqrt[3]{\frac{V_0}{2\pi}} = 0.042 \text{ m} = 1.65''$$
$$H = V_0/(\pi R^2) = 0.084 \text{ m} = 3.3''$$

Observation: If the goal is to design a hand-held soda-can, clearly the large radius is inappropriate. Proper ergonomic constraints must be placed; for example, a reasonable modification to Equation (2.29) is

$$\begin{aligned} &\underset{\{R,H\}}{\text{minimize}} && (2\pi R^2 + 2\pi RH) \\ &\text{s.t.} && \pi R^2 H - V_0 = 0 \\ & && R - R_{\max} \leq 0 \end{aligned} \tag{2.32}$$

It is indeed common to iterate on problem formulation until the solution is "satisfactory."

2.4 Analytical Design Problems

Next, we consider simple structural and engineering design problems where the objective and/or constraints can be expressed analytically.

Example 2.5 Point of Maximum Deflection

Problem: Consider a beam that is pinned at both ends, with an asymmetric load applied as illustrated in Figure 2.5; the Young's modulus is E and the moment of inertia is I. Find the point of maximum deflection; assume that $b < L/2$.

Modeling: This is, of course, a trivial problem in strength of materials where the beam deflection is known analytically; the deflection in the left segment is given by (see reference [10])

$$y(x) = \frac{Pb}{6EIL}[x^3 - (L^2 - b^2)x] \tag{2.33}$$

i.e.,

$$\underset{x}{\text{minimize}} \left(\frac{Pb}{6EIL}[x^3 - (L^2 - b^2)x] \right) \tag{2.34}$$

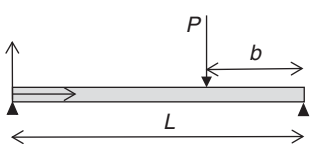

Figure 2.5 Beam deflection.

(cont.)

This is a single-variable, unconstrained minimization problem. Setting the derivative to zero, we have

$$\frac{dy}{dx} = \frac{Pb}{6EIL}[3x^2 - (L^2 - b^2)] = 0 \tag{2.35}$$

Thus,

$$x = \sqrt{\frac{(L^2 - b^2)}{3}} \tag{2.36}$$

Example 2.6 Spring Design

Problem: Next consider the classic spring design problem where the objective is to minimize the volume of a helical spring (see Figure 2.6) subject to deflection and stress constraints. The primary design variables are the number of coils (N), the outer diameter (D) and the wire diameter (d).

Modeling: Spring analysis is addressed in machine design handbooks such as reference [11]. The example is used here merely to highlight a typical design optimization problem.

The volume of a helical spring, after accounting for end-effects, can be approximated via

$$V = (N+2)\frac{\pi D}{2}\frac{\pi d^2}{4} = \pi^2\frac{(N+2)}{8}Dd^2 \tag{2.37}$$

Further, given an external load F, the deflection is given ([11]) by

$$\delta = \frac{8FD^3N}{d^4G} \tag{2.38}$$

where G is the shear modulus. Finally, the maximum shear stress can be approximated ([11]) by

$$\tau = \frac{8F(D/d)(1 + 0.5d/D)}{\pi d^2} \tag{2.39}$$

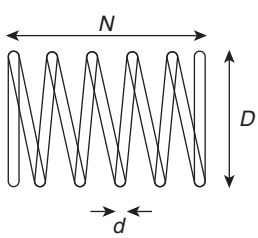

Figure 2.6 Helical spring.

(cont.)

Thus, the optimization problem can be posed as

$$\begin{aligned} \underset{\{N,D,d\}}{\text{minimize}} \quad & \pi^2 \frac{(N+2)}{8} D d^2 \\ \text{s.t.} \quad & \frac{8FD^3N}{d^4G} \leq \delta_{\max} \\ & \frac{8F(D/d)(1+0.5d/D)}{\pi d^2} \leq \tau_{\max} \end{aligned} \tag{2.40}$$

where the constraint limits are provided by the user. This is a *multi-variable, multi-constrained minimization* problem. In practice, additional constraints on the ratio D/d and the number of coils will be needed.

Solution: Since the objective and constraints are analytical functions of the design variables, such problems are computationally easy to solve using numerical algorithms to be discussed later.

2.5 Structural Analysis

Next, consider the problem illustrated in Figure 2.7, where a simple truss bridge is pinned at two ends, and a load is applied in the middle. Assume that all material and cross-sectional properties are given. The objective is to find the displacement at the point of load application.

At first glance, this does not appear to be an optimization problem, i.e., we are not trying to optimize any objective function. It is a structural *analysis* problem. However, there is an intricate relationship between structural analysis and optimization. This is captured by the *principle of minimum potential energy*.

The principle of minimum potential energy states that *a structural system* (such as the one in Figure 2.7) *subject to an external force will come to rest when its potential energy is a minimum* [12]. The potential energy is defined as

$$\Pi = U - 2W \tag{2.41}$$

where U is the internal elastic energy and W is the *quasi-static* work done by the force.

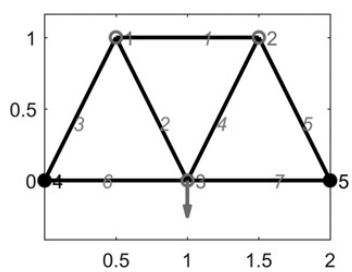

Figure 2.7 A truss analysis problem.

In other words, structural analysis is really an optimization problem. This duality between structural analysis and optimization is fascinating and important. Using the above principle, one can pose the problem of computing the displacement as a minimization problem. This is illustrated below through a few examples.

Example 2.7 Tensile Bar Analysis

Problem: Consider a tensile bar subject to an external force on one end and pinned at the other end (Figure 2.8). The properties of the tensile bar are as follows: E is the Young's modulus, A is the cross-sectional area and L is the length. Compute the deformation of the bar by minimizing its potential energy.

Modeling: Once again, observe that this is an analysis problem. To pose this as an optimization problem, we will appeal to the principle of minimum potential energy. The potential energy consists of two terms: elastic energy and work done. If the tensile bar undergoes a deformation δ, then its elastic energy is given by [10]

$$U = \frac{1}{2}k\delta^2 \tag{2.42}$$

where

$$k = \frac{EA}{L} \tag{2.43}$$

is the stiffness of the bar. On the other hand, the quasi-static work done is (see reference [10])

$$W = \frac{1}{2}P\delta \tag{2.44}$$

Observe the "1/2" in Equation (2.44); this arises because of the quasi-static nature of the force, i.e., the force is gradually increased from zero to the maximum value, rather a full force applied instantaneously. Thus, the potential energy is given by

$$\Pi(\delta) = U - 2W = \frac{1}{2}k\delta^2 - P\delta \tag{2.45}$$

Observe that the potential energy is a function of the unknown deformation δ. Since, at equilibrium, the potential energy takes a minimum, we differentiate Equation (2.45) with respect to δ, and set it equal to zero, leading to

Figure 2.8 A tensile bar problem.

(cont.)

$$\frac{d\Pi}{d\delta} = k\delta - P = 0 \tag{2.46}$$

i.e.,

$$\delta = P/k \tag{2.47}$$

Observe that Equation (2.46) is essentially the force balance equation, i.e., we have arrived at the force balance equation by differentiating the potential energy equation and setting it equal to zero! This *fundamental duality* is explored further in later chapters.

Example 2.8 Truss Analysis

Problem: Consider the truss system in Figure 2.9, subject to an external force. Find the deflection of the free node by minimizing its potential energy (the deflection is illustrated schematically by dashed lines). A and l are respectively the cross-sectional area and length of each bar.

Modeling: Let the displacement of the free node in the horizontal and vertical directions be u and v respectively. Assuming small displacements, using simple vector calculus one can show that the two bars undergo deformations of

$$\delta_1 \approx u\cos\theta - v\sin\theta \tag{2.48}$$

$$\delta_2 \approx -u\cos\theta - v\sin\theta \tag{2.49}$$

(see Exercise 2.11 at the end of this chapter). Thus, the total elastic energy is

$$U = \frac{1}{2}k(\delta_1)^2 + \frac{1}{2}k(\delta_2)^2 \tag{2.50}$$

where k captures the stiffness of the truss bar and is given by

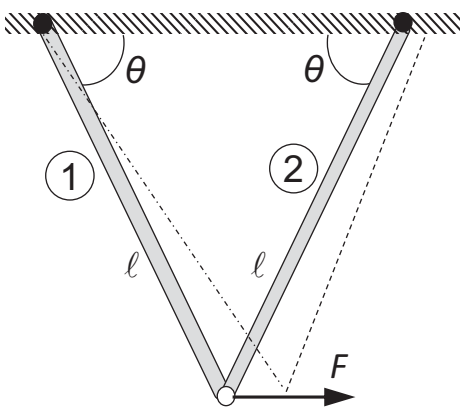

Figure 2.9 A two-bar truss analysis.

(cont.)

$$k = \frac{EA}{l} \tag{2.51}$$

From Equations (2.48) and (2.49), we have

$$U = \frac{1}{2}k(u\cos\theta - v\sin\theta)^2 + \frac{1}{2}k(-u\cos\theta - v\sin\theta)^2 \tag{2.52}$$

Simplifying,

$$U = k(u^2\cos^2\theta + v^2\sin^2\theta) \tag{2.53}$$

Further, since a horizontal force is applied, only the horizontal displacement plays a role in the quasi-static work done:

$$W = \frac{1}{2}Fu \tag{2.54}$$

Once again, the factor of "1/2" is due to the quasi-static nature of the force. Finally, the potential energy is given by

$$\Pi = U - 2W = k(u^2\cos^2\theta + v^2\sin^2\theta) - Fu \tag{2.55}$$

As stated earlier, the structural system will come to rest when the potential energy is a minimum. Thus, we pose the optimization problem as

$$\underset{\{u,v\}}{\text{minimize}}\ \Pi = k(u^2\cos^2\theta + v^2\sin^2\theta) - Fu \tag{2.56}$$

Solution: This is a single-objective, two-variable, unconstrained minimization problem. For multi-variable, unconstrained problems, we set the partial derivative of the objective with respect to each variable to zero, i.e.,

$$\frac{\partial[k(u^2\cos^2\theta + v^2\sin^2\theta) - Fu]}{\partial u} = 0 \tag{2.57}$$

$$\frac{\partial[k(u^2\cos^2\theta + v^2\sin^2\theta) - Fu]}{\partial v} = 0 \tag{2.58}$$

This results in

$$\begin{aligned} u &= F/(2k\cos^2\theta) = Fl/(2EA\cos^2\theta) \\ v &= 0 \end{aligned} \tag{2.59}$$

Again, the reader can verify that we will arrive at the same result via force balance.

Example 2.9 Spring System Analysis

Problem: Consider the spring system in Figure 2.10, consisting of two springs (marked as 1 and 2) that are fixed at top and bottom (nodes 2 and 3). An external force is applied at the middle node (node 1). The lengths of the two springs are 1 unit each (in the undeformed state), while their stiffnesses are 100 and 50 units respectively. The force components are 10 units in the x-direction and 8 units in the y-direction. Find the deflection of node 1 by minimizing the potential energy of the spring system.

(cont.)

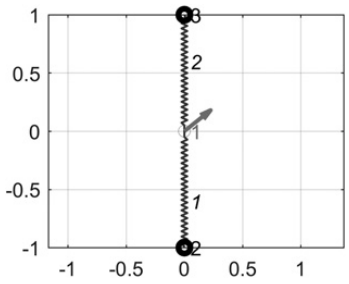

Figure 2.10 A spring system.

Modeling: Let the displacements of node 1 in the horizontal and vertical directions be u and v respectively. Observe that the undeformed length of each spring is 1 unit. Further, given an arbitrary displacement (u, v) of node 1, the deformed lengths of the two springs are

$$\begin{aligned} L_1 &= \sqrt{(0-u)^2 + (-1-v)^2} = \sqrt{u^2 + (1+v)^2} \\ L_2 &= \sqrt{(0-u)^2 + (1-v)^2} = \sqrt{u^2 + (1-v)^2} \end{aligned} \tag{2.60}$$

Since the deformation can be fairly large, one cannot make the truss-based simplification. Therefore, the increase in length is given by

$$\begin{aligned} \Delta L_1 &= \sqrt{u^2 + (1+v)^2} - 1 \\ \Delta L_2 &= \sqrt{u^2 + (1-v)^2} - 1 \end{aligned} \tag{2.61}$$

This results in an elastic energy of

$$\begin{aligned} U_1 &= 0.5k_1(\Delta L_1)^2 = 0.5k_1\left(\sqrt{u^2 + (1+v)^2} - 1\right)^2 \\ U_2 &= 0.5k_2(\Delta L_2)^2 = 0.5k_2\left(\sqrt{u^2 + (1-v)^2} - 1\right)^2 \end{aligned} \tag{2.62}$$

Given the spring stiffnesses of 100 and 50 units, the total elastic energy is given by

$$U = 0.5\begin{bmatrix} 100\left(\sqrt{u^2 + (1+v)^2} - 1\right)^2 \\ +\,50\left(\sqrt{u^2 + (1-v)^2} - 1\right)^2 \end{bmatrix} \tag{2.63}$$

The external work done is

$$W = (1/2)(f_x u + f_y v) = (1/2)(10u + 8v) \tag{2.64}$$

Finally, the potential energy is given by

$$\Pi = 0.5\begin{bmatrix} 100\left(\sqrt{u^2 + (1+v)^2} - 1\right)^2 \\ +\,50\left(\sqrt{u^2 + (1-v)^2} - 1\right)^2 \end{bmatrix} - (10u + 8v) \tag{2.65}$$