Paul E. Fishback

Linear and Nonlinear Programming with Maple: An Interactive, Applications-Based Approach

For Barb, Andy, and Margaret

Contents

List of Figures

List of Tables

Foreword

This book is designed for use as a primary text in an introductory course, or sequence of courses, on linear and nonlinear programming. Its intended audience consists of undergraduate students who have completed both a standard single-variable calculus sequence, along with an introductory linear algebra course. Although some background knowledge of multivariable calculus (primarily partial derivatives) and some experience with formal proof writing are helpful, they are by no means essential.

The book itself is organized into two parts, with the first focusing on linear programming and the second on nonlinear programming. Following these two parts are four appendices, which contain course projects, Maple resources, and a summary of important linear algebra facts.

Focus on Linear Algebra

A primary goal of this book is to "bridge the gap," so to speak, which separates the two primary classes of textbooks on linear and nonlinear programming currently available to students. One consists of those management science books that lack the level of mathematical detail and rigor this text seeks to attain. Typically, they assume little to no linear algebra background knowledge on the part of the reader. Texts in the second class are better suited for graduate level courses on mathematical programming. In simple terms, they are written at "too high a level" for this book's intended audience.

Undergraduate students who use this book will be exposed early to topics from introductory linear algebra, such as properties of invertible matrices and facts regarding nullspaces. Eigenvalues, of course, are an essential for the classification of quadratic forms. Most important, however, is the extent to which partitioned matrices play a central role in developing major ideas. In particular, the reader discovers in Section 2.4 that the Simplex Algorithm may be viewed entirely in terms of multiplication of such matrices. This perspective from which to view the algorithm provides streamlined approaches for constructing the Revised Simplex Method, developing duality theory, and approaching the process of sensitivity analysis.

Some linear algebra topics arising in this text are not ones usually encountered by students in an introductory course. Most notable are certain properties of

the matrix transpose, the Spectral Theorem, and facts regarding matrix norms. As these topics arise in the text, brief digressions summarize key ideas, and the reader is referred to appropriate locations in the appendices.

Maple

As the title indicates, Maple is the software of choice for this text. While many practitioners in the field of mathematical programming do not consider Maple well-suited for large-scale problems, this software is ideal in terms of its ability to meet the pedagogical goals of this text and is accessible to students at over 90% of advanced research institutions worldwide. By combining Maple's symbolic computing components, its numeric capabilities, its graphical versatility, and its intuitive programming structures, the student who uses this text should acquire a deep conceptual understanding of major mathematical programming principles, along with the ability to solve moderately-sized real world applications.

The text does not assume prior Maple experience on the part of its reader. The Maple novice should first read Appendix B, Getting Started with Maple. It provides a sufficient amount of instruction for the reader unfamiliar with this software, but no more than that needed to start reading Chapter 1. Maple commands are then introduced throughout the book, as the need arises, and a summary of all such commands is found in Appendix 4. Finally, sample Maple work sheets are provided in the text itself and are also accessible online at `www.lp-nlp-with-maple.org`.

Waypoints

Any mathematics text should strive to engage its reader. This book is no exception, and, hopefully, it achieves success in this regard. Interspersed throughout are "Waypoints," where the reader is asked to perform a simple computation or to answer a brief question, both of which are designed to assess his or her understanding. The intent of these Waypoints is to facilitate the "hands-on," or interactive, learning approach suggested by this book's title. The instructor can utilize them in various ways, for example, by using them as a means to intersperse lecture with small-group discussions or as an additional source of text exercises.

Projects

In addition to Waypoints and numerous exercises, both computational and conceptual in nature, this text includes six substantial projects, which are given in Appendix A. Three of the six focus on linear programming models: Two transportation-related problems and one integer linear programming problem on work scheduling. The remaining three focus on nonlinear programming: A linear classifier problem for cancer diagnosis, the Markowitz

portfolio model, and a bimatrix game theory representation of a predator-prey habitat. These six projects provide the student opportunities to investigate real-world applications more substantial than those found in the exercises. They are well-suited for use in a class in which students complete team projects designed to reinforce their problem solving and mathematical communication skills. Maple is indispensable for addressing the problems posed by the projects, and students can easily produce reports consisting of Maple work sheets that combine the software's text- and math-input environments.

Possible Course Outlines

There are several different possible outlines for using this book as a primary course text, depending upon course goals and instructional time devoted to the various topics. In terms of linear programming, core topics include Chapter 1, Sections 2.1-2.4, Chapter 3, Sections 4.1-4.2, and Section 5.1. These cover the essentials of linear programming, basic applications, duality, sensitivity analysis, and integer linear programming via the branch and bound method. Sections 2.5, 4.3, and 5.2 address optional topics, namely the interior point algorithm, the dual simplex method, and the cutting plane algorithm, respectively. When considering which of these three additional topics to cover, however, the instructor should bear in mind that the cutting plane algorithm requires use of the dual simplex method.

For a follow-up course on nonlinear programming, core topics include Chapter 6, Sections 7.1-7.2, and Sections 8.1-8.2. These cover the basics of unconstrained optimization, the Steepest Descent Method, Newton's Method, the Karush-Kuhn-Tucker Necessary Conditions Theorem, and convex nonlinear programming. Optional topics found in Sections 7.3, 8.3-8.5 include the Levenberg-Marquardt Algorithm, saddle point criteria, quadratic programming, and sequential quadratic programming.

Acknowledgements

I am extremely grateful to the many individuals who provided me assistance as I wrote this book. Not having previously undertaken such a task, I sought general advice from numerous colleagues: Edward Aboufadel, Matthew Boelkins, Jonathan Hodge, and Theodore Sundstrom (all at Grand Valley State University); Larry Knop (Hamilton College); Anne Young (Loyola College of Maryland). Their advice, along with the guidance of Bob Stern from Taylor and Francis, made the process of completing this text run smoothly.

Numerous students and colleagues helped me improve specific portions of the manuscript as I piloted it in classes at Grand Valley State University. More than anyone, my colleague Clark Wells has been a valuable source of

information and advice, both in terms of writing this book and also with regards to teaching mathematical programming in general.

My family exhibited enormous patience over the four-year period I devoted to writing this text, and they are likely quite relieved that the task is complete. Finally, I am deeply indebted to Dr.'s Kost Elisevich, Brien Smith, and Timothy Thoits.

Part I

Linear Programming

Chapter 1

An Introduction to Linear Programming

1.1 The Basic Linear Programming Problem Formulation

Any student who has completed a first-semester calculus course recalls solving optimization problems. A contrived, yet typical example of such is given by the following:

> Farmer Smith has 100 yards of fence to build a rectangular corral, one of whose sides will consist of the side of a barn. What should be the dimensions of the corral if Farmer Smith wishes to maximize the possible enclosed area?

Of course, the calculus provides one means of solving this problem. If we let x and y denote the dimensions of the corral, then the available fence length dictates $2x + 2y = 100$, and the area, A, is given by $A = xy$. Rewriting A as a function of a single variable, say x, and solving the equation, $\frac{dA}{dx} = 0$, for x, we deduce that Farmer Brown maximizes the enclosed area by using 50 feet of fence opposite the barn and 25 feet for each of the opposite two sides.

Linear programming, the main focus of this chapter, is another optimization tool having components analogous to those of the preceding calculus problem. First, there is a function to be maximized or minimized. Whereas in calculus, one works with smooth functions of a single variable, linear programming involves optimizing linear functions of several variables. In calculus, one frequently constrains the input values of the function to fall within a certain interval. In linear programming, input values must satisfy several constraints, each of which takes the form of a linear inequality. Calculus optimization problems rely upon the derivative; linear programming utilizes linear algebra as a tool. The goal of Part I of this text is to expand upon these ideas and to apply them to solve a variety of real-world linear programming problems.

1.1.1 A Prototype Example: The Blending Problem

One sector of the economy that has made extensive use of linear programming is the petroleum industry. Texaco, for example, used linear programming to address *blending problems* in the early 1980s; doing so saved the company an estimated $30 million per year [11]. The idea of a blending problem is fairly simple. A company produces various grades of gasoline, e.g., regular unleaded, mid-grade, and premium, which have various octane ratings. Various stocks are used to produce these grades. Such stocks are the intermediate products produced by the refineries and include straight-run gasoline, formate, and catalytic gasoline. The question becomes, given a particular quantity the company wishes to optimize, e.g., revenue earned on the various gasoline grades, how does it do so while satisfying numerous constraints on stock demand and quality specifications?

A fictitious, yet simple and illuminating example used throughout this chapter is that of the *FuelPro Petroleum Company*, a small refinery that sells two grades of fuel, premium and regular unleaded. For purposes of simplicity, we will assume that only two stocks, stock A and stock B, are used to produce the two grades and that 28 and 32 gallons of each stock, respectively, are available. The following table summarizes how much of each stock is required for each grade. (Assume all quantities are measured in gallons.)

TABLE 1.1: *FuelPro* production data

Stock	Premium	Reg. Unleaded	Available Stock
Type A	2	2	28
Type B	3	2	32

Production facility limitations dictate that at most 8 gallons of premium grade fuel can be made available each hour. The net profit per gallon of premium grade is $.40 and that of regular unleaded is $.30. *FuelPro* is fortunate to sell all the fuel it refines.

Some logical questions the company might wish to address are the following:

1. What are the possible production combinations of fuel types that will satisfy the above conditions or *set of constraints*?

2. What combinations of fuel types will *maximize FuelPro* 's profit?

3. How *sensitive* is the answer to the preceding question to the various parameters in the model? For example, by how much will the price of unleaded regular need to change before the optimal fuel production combination from (2) will change as well?

Linear programming is a powerful tool for addressing such questions!

Waypoint 1.1.1. In the preceding example, experiment and list four combinations of premium and regular unleaded fuel types, three of which satisfy the listed constraints and one of which does not. Of those three that do, which produces the largest profit?

Mere experimentation with points as a means of determining the combination of fuel types that maximizes profit for *FuelPro* naturally leads to the question of whether there is a mathematically more systematic approach to solving this problem. In this section we take a first step toward developing such an approach by introducing basic terminology and by interpreting the problem situation using the language of multivariate functions and linear algebra.

Let us denote the number of gallons of premium grade and regular unleaded produced in a given hour by the variables x_1 and x_2, respectively. Using information regarding the production of the two fuel grade types, we can construct inequalities relating these two variables.

1. From information provided in the first row of Table 1.1, we know that each gallon of premium requires 2 gallons of stock A and each gallon of regular unleaded requires 2 gallons of stock A. Since only 28 gallons of stock A are available to *FuelPro* each hour, we obtain the inequality:

$$2x_1 + 2x_2 \leq 28.$$

2. By similar reasoning, the maximum availability of stock B dictates that

$$3x_1 + 2x_2 \leq 32.$$

3. At most 8 gallons of premium grade fuel can be sold each hour. Thus $x_1 \leq 8$.

4. Only nonnegative values of x_1 and x_2 make sense for the problem situation. Therefore, $x_1, x_2 \geq 0$.

FuelPro desires to maximize its profit subject to the preceding constraints. Given the profit per gallon amounts for the premium and unleaded fuels, the function to be maximized is simply $.4x_1 + .3x_2$. For the mere sake of working with integer-valued coefficients, we shall instead maximize

$$z = f(x_1, x_2) = 4x_1 + 3x_2.$$

Combining this function with the above constraints, the optimization problem faced by *FuelPro* can be written as follows:

$$\begin{aligned} &\text{maximize } z = f(x_1, x_2) = 4x_1 + 3x_2 \\ &\text{subject to} \\ &\quad x_1 \leq 8 \\ &\quad 2x_1 + 2x_2 \leq 28 \\ &\quad 3x_1 + 2x_2 \leq 32 \\ &\quad x_1, x_2 \geq 0. \end{aligned} \tag{1.1}$$

System (1.1) is an example of a *linear programming problem* (LP).

Definition 1.1.1. A linear programming problem is a mathematical problem that consists of the following components:

1. An *objective function*, f, of n *decision variables*, $x_1, x_2, \ldots, x_n$, that is to be maximized or minimized. This function is linear; that is it can be written in the form

$$z = f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n, \tag{1.2}$$

   where each c_i belongs to $\mathbb{R}$.

2. A set of m *constraints* or inequalities. Each constraint is linear in that it takes the form,

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i, \tag{1.3}$$

   where $1 \leq i \leq m$ and where each a_{ij} and b_i belongs to $\mathbb{R}$.

3. Possible *sign restrictions* placed on any of the decision variables.

We will define the vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ in $\mathbb{R}^n$ to be a *feasible solution* of the LP if it satisfies all constraints and sign restrictions of the LP. The *feasible region* is the set of all feasible solutions, and an *optimal solution* is a feasible solution whose corresponding objective function value is greater than or equal to that of any other feasible solution for a maximization problem and less than or equal to that of any other feasible solution for a minimization problem.

While Definition 1.1.1 appears to stipulate that all constraints are of inequality type, equations are permissible as well, as long as the constraint equations are linear in the variables $x_1, x_2, \ldots, x_n$. However, in Section 1.1.3 we will discover how the set of constraints of any LP can be expressed entirely in terms of inequalities.

It is important to note that any LP has hidden assumptions due to its linear nature. For example, in our petroleum model we assume that there is no interaction between the amounts of premium and regular unleaded, e.g., no product terms x_1x_2. We are also assuming that the objective function is linear. In certain other situations, the function f might be better modeled by a quadratic function, in which case the problem is not an LP but a quadratic programming problem. Finally, we are assuming that the decision variables can take on fractional, noninteger values, which may not always be realistic for the application at hand. In Chapter 5 we discuss problems in which we desire only integer-valued solutions for one or more of our decision variables.

1.1.2 Maple's LPSolve Command

Maple's `Optimization` package provides a convenient means for solving an LP such as (1.1). A simple worksheet for doing so is given as follows:

```
> restart;
> with(Optimization);
```

[*ImportMPS*, *Interactive*, *LPSolve*, *LSSolve*, *Maximize*, *Minimize*, *NLPSolve*, *QPSolve*]

```
> f:=(x1,x2)->4*x1+3*x2;
```

$$f := (x1, x2) \rightarrow 4x_1 + 3x_2$$

```
> constraints:=[x1<=8,2*x1+2*x2<=28,3*x1+2*x2<=32];
```

$$constraints := [x_1 \leq 8, 2x_1 + 2x_2 \leq 28, 3x_1 + 2x_2 \leq 32]$$

```
> LPSolve(f(x1,x2),constraints,'maximize', assume=nonnegative);
```

$$[46., [x_1 = 4.000000000, x_2 = 9.999999999]]$$

Maple's `LPSolve` command yields a floating-point representation of the optimal solution, $x_1 = 4$ and $x_2 = 10$, which corresponds to an objective value $z = f(4, 10) = 46$. Hence, *FuelPro* should produce 4 gallons of premium grade and 10 gallons of regular unleaded each hour in order to maximize its profit, which equals $ 4.60.

The general form of the `LPSolve` command is given by

`LPSolve(`*`objective function`*`,`*`list of constraints`*`,`*`options`*`)`.

If no options are given, Maple minimizes the objective function by default and makes no assumptions regarding sign values of decision variables. In the preceding example, we have added options that maximize the objective function and require nonnegative decision variables. Doing so guarantees

that `LPSolve` obtains a solution coinciding with that of (1.6). In subsequent sections, we will elaborate on features of this command and how it can be used to solve more complicated problems.

In the *FuelPro* example, the LP has a unique optimal solution, $x_1 = 4$ and $x_2 = 10$. As we shall discover in Section 1.2, the solution of any LP takes one of four general forms:

1. The LP can be *infeasible*, meaning that its feasible region is the empty set.
2. The LP can have a *unique optimal solution*.
3. The LP can have more than one optimal solution, in which case we say that it has *alternative optimal solutions*.
4. The LP can be *unbounded*. For a problem in which one seeks to maximize the objective function, this means that the objective function can be made as large as one likes. For a minimization problem, this means that the objective function can be made as small as one likes.

When an LP is infeasible, Maple's `LPSolve` command returns an error message, and when the LP is unbounded, Maple returns a feasible solution along with a warning message. When an LP has more than one solution, Maple returns one solution, overlooking the others.

1.1.3 The Matrix Inequality Form of an LP

Matrix notation provides a convenient means for restating a linear programming problem such as (1.1) in more compact form. Let us denote the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 2 \\ 3 & 2 \end{bmatrix}$$

and the vectors $\mathbf{c} = \begin{bmatrix} 4 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}$, and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Then (1.1) becomes

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} \le \mathbf{b} \\ &\qquad \mathbf{x} \ge \mathbf{0}. \end{aligned} \tag{1.4}$$

In this notation, a vector inequality such as $A\mathbf{x} \le \mathbf{b}$ means that the inequality is valid for every pair of corresponding entries in the vectors. We shall refer to (1.4) as the *matrix inequality form* of the LP.

Maple's `LPSolve` command accepts LP's expressed in matrix inequality form. For example (1.4), can be solved as follows:

```
> c:=Vector[row]([4,3]);
```

$$c := \begin{bmatrix} 4 & 3 \end{bmatrix}$$

```
> A:=Matrix(3,2,[1,0,2,2,3,2]);
```

$$A := \begin{bmatrix} 1 & 0 \\ 2 & 2 \\ 3 & 2 \end{bmatrix}$$

```
> b:=<8,28,32>;
```

$$b := \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}$$

```
> LPSolve(c, [A, b], assume = nonnegative,'maximize');
```

$$\begin{bmatrix} 46 & \begin{bmatrix} 4 \\ 10 \end{bmatrix} \end{bmatrix}$$

Observe that Maple's output consists of a list. The first entry is the optimal objective value; the second entry is the corresponding vector of decision variable values.

Through simple algebraic manipulations, we can express every LP in inequality form. For example, the LP given in (1.5) has constraints involving inequalities and equations.

$$\begin{aligned} &\text{maximize } z = f(x_1, x_2) = 2x_1 + 3x_2 \\ &\quad \text{subject to} \\ &\qquad x_1 + x_2 \le 6 \\ &\qquad x_1 + 5x_2 \ge 10 \\ &\qquad 2x_1 - x_2 = 3 \\ &\qquad x_1, x_2 \ge 0. \end{aligned} \tag{1.5}$$

The second constraint can be rewritten as $-x_1 - 5x_2 \le -10$. The third is the combination of two constraints, $2x_1 - x_2 \le 3$ and $-2x_1 + x_2 \le -3$. Thus, (1.5) becomes

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\quad \text{subject to} \\ &\qquad A\mathbf{x} \le \mathbf{b} \\ &\qquad \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

where $A = \begin{bmatrix} 1 & 1 \\ -1 & -5 \\ 2 & -1 \\ -2 & 1 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 2 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 6 \\ -10 \\ 3 \\ -3 \end{bmatrix}$, and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

Exercises in this section demonstrate other means for expressing LPs in matrix inequality form, including those containing variables unrestricted in sign or those whose constraints involve absolute values.

Exercises Section 1.1

1. Express each LP below in matrix inequality form. Then solve the LP using Maple provided it is feasible and bounded.

   (a)

$$\begin{aligned} \text{maximize } z &= 6x_1 + 4x_2 \\ \text{subject to} & \\ 2x_1 + 3x_2 &\le 9 \\ x_1 &\ge 4 \\ x_2 &\le 6 \\ x_1, x_2 &\ge 0 \end{aligned}$$

   (b)

$$\begin{aligned} \text{maximize } z &= 3x_1 + 2x_2 \\ \text{subject to} & \\ x_1 &\le 4 \\ x_1 + 3x_2 &\le 15 \\ 2x_1 + x_2 &= 10 \\ x_1, x_2 &\ge 0 \end{aligned}$$

   (c)

$$\begin{aligned} \text{maximize } z &= -x_1 + 4x_2 \\ \text{subject to} & \\ -x_1 + x_2 &\le 1 \\ x_1 + &\le 3 \\ x_1 + x_2 &\ge 5 \\ x_1, x_2 &\ge 0 \end{aligned}$$

(d)

$$\begin{aligned} \text{minimize } z &= -x_1 + 4x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\geq 5 \\ x_1 + x_2 &\geq 4 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

(Hint: First change the LP so that the goal is "maximize" $\tilde{z} = -z = x_1 - 4x_2$.)

(e)

$$\begin{aligned} \text{maximize } z &= 2x_1 - x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\geq 8 \\ x_1 + x_2 &\geq 4 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

(f)

$$\begin{aligned} \text{minimize } z &= 2x_1 + 3x_2 \\ \text{subject to} & \\ 3x_1 + x_2 &\geq 1 \\ x_1 + x_2 &\leq 6 \\ x_2 &\geq 0 \end{aligned}$$

(Hint: The variable x_1 is unrestricted in sign. Define $x_1 = x_{1,+} - x_{1,-}$, where $x_{1,+}$ and $x_{1,-}$ are nonnegative. Express the LP in terms of the three nonnegative decision variables, x_2, $x_{1,+}$ and $x_{1,-}$.)

2. In certain situations an optimization problem may not appear to be an LP, such as the case when constraints involve certain absolute value inequalities. For example, consider the following problem:

$$\begin{aligned} \text{minimize } z &= x_1 + 4x_2 \\ \text{subject to} & \\ x_1 + 2x_2 &\leq 5 \\ |x_1 - x_2| &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

By restating the absolute value constraint as a combination of two linear constraints, show that this problem is in fact an LP. Write the LP in matrix inequality form, and determine its solution using Maple.

3. Congratulations! Upon graduating from college, you've immediately been offered a high-paying position as president of the Lego Furniture Company.[1] Your company produces chairs (each requiring 2 square blocks and 1 rectangular block) as well as tables (each requiring 2 square blocks and 2 rectangular blocks) and has available resources consisting of 8 rectangular blocks and 6 square ones. Assume chairs and tables each sell for $5 and $7, respectively, and that your company sells all of what it produces.

   (a) Set up an LP whose objective is to maximize your company's revenue. Solve this LP through trial and error, using your Legos if you wish.

   (b) Now solve the LP with the aid of Maple.

4. (*The Foraging Herbivore Model*) In certain parts of the U.S. the vole, or common field mouse, is an herbivore whose diet consists predominantly of grass and a type of broad-leafed herb known as forb. Research suggests that the vole searches for food in a manner that minimizes its total foraging time, subject to a set of two constraints. [2]

   Table 1.2 summarizes grass and forb data regarding digestive capacity and energy content. Food bulk records the extent to which the mass of a substance increases after it enters the digestive system and becomes liquid-saturated. For example, two grams of grass, when consumed, expands to $2 \times 1.64 = 3.28$ grams within the digestive system. To distinguish the mass of food prior to consumption from that in the digestive system, we use units of gm-dry and gm-wet, respectively.

TABLE 1.2: Dietary data for vole

	Food bulk (gm-wet/gm-dry)	Energy content (kcal/gm)
Grass	1.64	2.11
Forb	2.67	2.30

   The digestive capacity of the vole is 31.2 gm-wet per day, and the vole must consume enough food to meet an energy requirement of at least 13.9 kcal per day. Assume that the vole's foraging rate is 45.55 minutes per gram of grass and 21.87 minutes per gram of forb.

   (a) Let x_1 and x_2 denote the number of grams of grass and number of grams of forb, respectively, consumed by the vole on a given day.

[1] Based upon Pendegraft, [37], (1997). For this problem it helps to have some Legos, specifically 8 large rectangular blocks and 6 small square blocks.

[2] Based upon Belovsky, [5], (1984).

Construct an LP that minimizes the vole's total daily foraging time given the digestive capacity and energy constraints. Verify that your units make sense for each constraint and for your objective function. Then write this LP in matrix inequality form.

(b) Solve the LP using Maple.

1.2 Linear Programming: A Graphical Perspective in $\mathbb{R}^2$

In Section 1.1 we formulated the *FuelPro* problem and obtained its solution with the aid of Maple's `Optimization` package. Now we develop a graphical approach for obtaining the same solution, one that provides deeper insight into the nature of the solutions to LPs in general. Recall the *FuelPro* problem:

$$\begin{aligned} &\text{maximize } z = f(x_1, x_2) = 4x_1 + 3x_2 \\ &\quad \text{subject to} \\ &\qquad x_1 \leq 8 \\ &\qquad 2x_1 + 2x_2 \leq 28 \\ &\qquad 3x_1 + 2x_2 \leq 32 \\ &\qquad x_1, x_2 \geq 0. \end{aligned} \tag{1.6}$$

The points in the x_1x_2-plane that satisfy all the constraints and sign conditions in (1.6) form the *feasible region* for the *FuelPro* model.

Waypoint 1.2.1. Sketch and shade the graph of the feasible region in the x_1x_2-plane. While this task is easily accomplished by hand, Maple can also do so through the use of the `inequal` command as described in Appendices C and D. The list of inequalities used within this command is given by `[x1<=8, 2*x1+2*x2<=28, 3*x1+2*x2<=32,x1>=0,x2>=0 ]` . The resulting graph should resemble that in Figure 1.1

The function $f = 4x_1 + 3x_2$ from (1.6) is a multivariate function whose graph consists of a surface in $\mathbb{R}^3$. Recall that a *contour* or *level curve* of such a function is a curve in the x_1x_2-plane that results from holding z constant. In other words, it is a curve that results from slicing the surface with a horizontal plane of the form $z = c$, where c is a constant. A *contour diagram* is merely a collection of contours plotted on a single set of axes.

Maple produces contour diagrams through its `contourplot` command, located in the `plots` package. The general form of the command is `contourplot(`*`expression in x1 and x2`*`,`*`x1`*`=a..b,`*`x2`*`=c..d,`*`options`*`)`. Various options include specifying the output values used for the contours, specifically via `contours=L`, where `L` is a list of contour values. Figure 1.2 illustrates an example, using the expression $x_1^2 + x_2^2$, along with contour values of 0, 1, 2, 3, and 4:

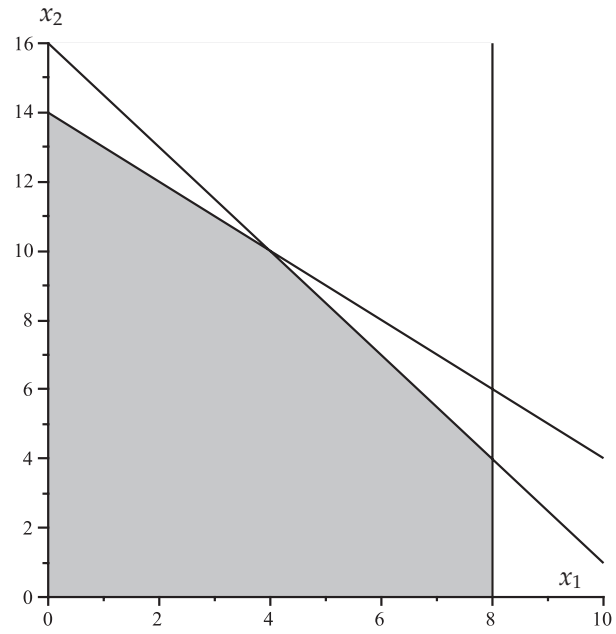


FIGURE 1.1: Feasible region for *FuelPro* LP.

```
> contourplot(x1^2+x2^2,x1=-3..3,x2=-3..3,contours=[1,2,3,4]);
```

For the *FuelPro* problem, the objective function f is a *linear function* of two variables so that its contours form *parallel lines*.

◆ ◆

Waypoint 1.2.2. Superimpose on your feasible region from the previous Waypoint the graphs of several contours of the objective function $f(x_1, x_2) = 4x_1 + 3x_2$. (Recall from Appendices C and D that plot structures can be superimposed by use of the `display` command.) Then use these contours to determine the amounts of premium and regular unleaded fuel types in the feasible region that correspond to the maximum profit. Your solution should of course agree with that obtained by using the `LPSolve` command in Section 1.1.

◆ ◆

Recall from Section 1.1 that the solution of an LP can take one of four possible forms.

1. The LP can be *infeasible*, meaning that its feasible region is the empty set.

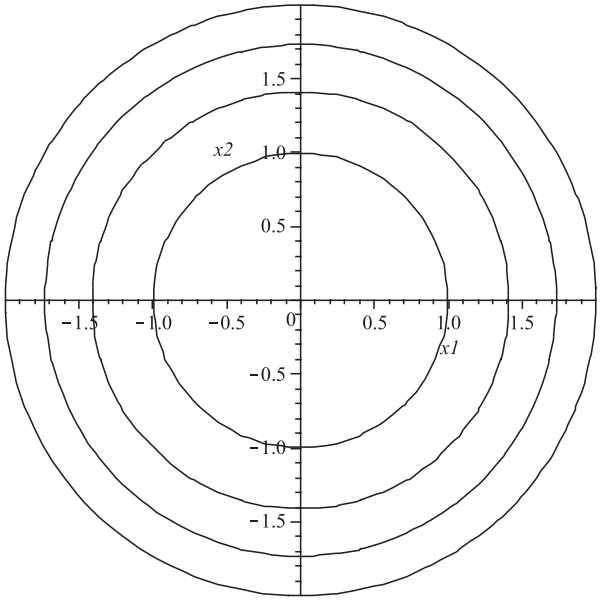


FIGURE 1.2: Sample contour diagram produced by Maple.

2. The LP can have a *unique optimal solution*, as is the case for the *FuelPro* problem.

3. The LP can have *alternative optimal solutions*. That is, there exists at least two optimal solutions.

4. The LP can be *unbounded*. For a problem in which one seeks to maximize the objective function, this means that the objective function can be made as large as one likes. For a minimization problem, this means that the objective function can be made as small as one likes.

While constraints and sign conditions alone determine whether an LP is infeasible, contours provide a graphical means for determining whether an LP has alternative optimal solutions or is unbounded. The LP in (1.7) provides one such example.

$$\begin{aligned} &\text{maximize } z = f(x_1, x_2) = 6x_1 + 4x_2 \\ &\quad \text{subject to} \\ &\quad 3x_1 + 2x_2 \leq 18 \\ &\quad x_1 \qquad \leq 4 \\ &\quad \qquad x_2 \leq 6 \\ &\quad x_1, x_2 \geq 0. \end{aligned} \tag{1.7}$$

The feasible region for (1.7), together with objective function contours $z = 32$, $z = 34$ and $z = 36$, is shown in Figure 1.3.

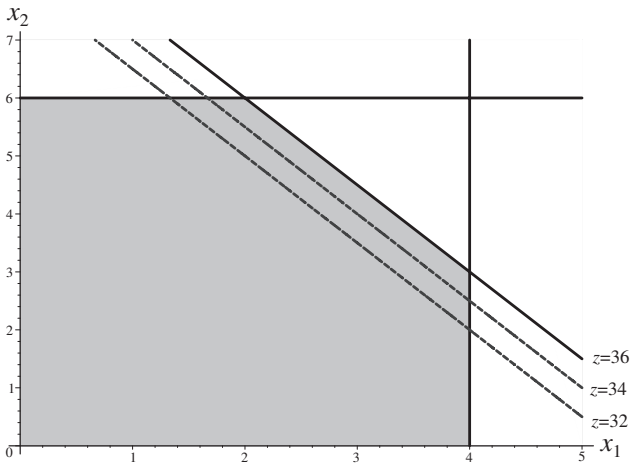


FIGURE 1.3: LP with alternative optimal solutions.

In this example, contours of the objective function f are parallel to the boundary of the constraint $3x_1 + 2x_2 \leq 9$. Moreover, objective values corresponding to these contours illustrate how all points on the segment from $(2, 6)$ to $(4, 3)$ yield the same optimal objective value, $z = 36$. One way to represent this segment is through a parametric representation as follows:

$$(x_1, x_2) = t \cdot (2, 6) + (1 - t) \cdot (4, 3) = (4 - 2t, 3 + 3t), \text{ where } 0 \leq t \leq 1.$$

The preceding example illustrates how contours can be used to identify an LP having alternative optimal solutions. Namely, such solutions arise when the contour that intersects the feasible region and corresponds to the optimal objective value also contains a segment of the feasible region boundary. That contours can also be used to identify unbounded LPs is left as an exercise.

Exercises Section 1.2

1. For each LP below, graph the feasible region, and sketch at least three contours of the objective function. Use your contours to determine the nature of the solution. If the LP has a unique optimal solution, list the values of the decision variables along with the corresponding objective

function value. If the LP has alternative optimal solutions, express the general solution in parametric form. You may wish to combine Maple's `inequal`, `contourplot`, and `display` commands to check each solution.

(a)

$$\begin{aligned} \text{maximize } z &= 6x_1 + 4x_2 \\ \text{subject to} & \\ 2x_1 + 3x_2 &\le 9 \\ x_1 &\ge 4 \\ x_2 &\le 6 \\ x_1, x_2 &\ge 0 \end{aligned}$$

(b)

$$\begin{aligned} \text{maximize } z &= 3x_1 + 2x_2 \\ \text{subject to} & \\ x_1 &\le 4 \\ x_1 + 3x_2 &\le 15 \\ 2x_1 + x_2 &= 10 \\ x_1, x_2 &\ge 0 \end{aligned}$$

(c)

$$\begin{aligned} \text{minimize } z &= -x_1 + 4x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\ge 5 \\ x_1 + x_2 &\ge 4 \\ x_1 - x_2 &\le 2 \\ x_1, x_2 &\ge 0 \end{aligned}$$

(d)

$$\begin{aligned} \text{maximize } z &= 2x_1 + 6x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\le 6 \\ x_1 + 2x_2 &\ge 5 \\ x_1, x_2 &\ge 0 \end{aligned}$$

(e)

$$\begin{aligned} \text{minimize } z &= 2x_1 + 3x_2 \\ \text{subject to} & \\ 3x_1 + x_2 &\geq 1 \\ x_1 + x_2 &\leq 6 \\ x_2 &\geq 0 \end{aligned}$$

2. Use the graphical methods discussed in this section to solve the *Foraging Herbivore Model*, Exercise 4, from Section 1.1.

3. Consider the LP given by

$$\begin{aligned} \text{maximize } z &= 2x_1 - x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\geq 8 \\ x_1 + x_2 &\geq 4 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

   (a) Sketch the feasible region for this LP, along with the contours corresponding to $z = 50$, $z = 100$, and $z = 150$.

   (b) If M is an arbitrarily large positive real number, what is a description, in terms of M, of the feasible points (x_1, x_2) corresponding to $z = M$.

   (c) Explain why the LP is unbounded.

4. Suppose f is a linear function of x_1 and x_2, and consider the LP given by

$$\begin{aligned} \text{maximize } z &= f(x_1, x_2) \\ \text{subject to} & \\ x_1 &\geq 1 \\ x_2 &\leq 1 \\ x_1, x_2 &\geq 0. \end{aligned}$$

   (a) Sketch the feasible region for this LP.

   (b) Give an example of an objective function f for which the LP has a unique optimal solution.

   (c) Give an example of an objective function f for which the LP is unbounded.

   (d) Give an example of an objective function f for which the LP has alternative optimal solutions.

1.3 Basic Feasible Solutions

The *FuelPro* LP, (1.1), has as its optimal solution, $x_1 = 4$ and $x_2 = 10$, which corresponds to an objective value of $z = 46$. The feasible region for the LP, labeled with the optimal solution, is shown in Figure 1.4.

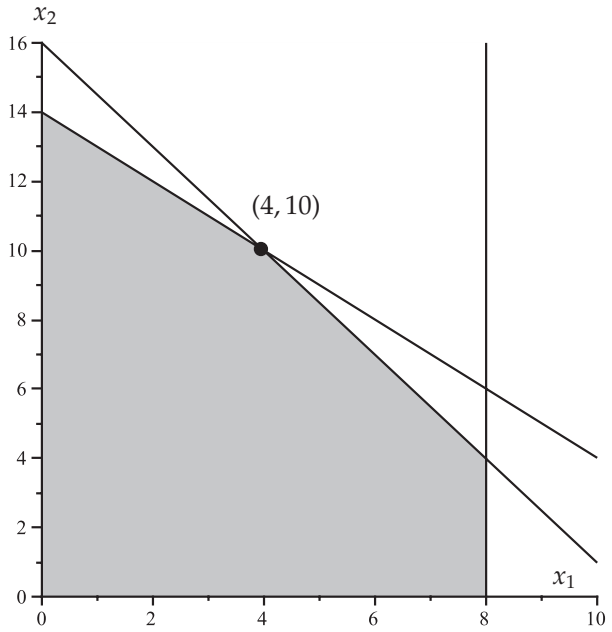


FIGURE 1.4: Feasible region for *FuelPro* LP.

We see that the feasible region has five "corner points" or "extreme points," one of which is the optimal solution. This result holds true in general. Namely, for the LP (1.4) having a unique optimal solution, the optimal solution occurs at an "extreme point" of the feasible region in $\mathbb{R}^n$. The *simplex algorithm*, which we develop in Section 2.1, is a linear-algebra based method that graphically corresponds to starting at one extreme point on the boundary of the feasible region, typically $(0, 0)$, and moving to adjacent extreme points through an iterative process until an optimal solution is obtained. The purpose of this section is to lay a foundation for this algorithm by developing appropriate terminology that connects the underlying ideas of systems of equations to those of the feasible region and its extreme points.

The *FuelPro* LP (1.1) is given by

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 \\ &\text{subject to} \\ &x_1 \leq 8 \\ &2x_1 + 2x_2 \leq 28 \\ &3x_1 + 2x_2 \leq 32 \\ &x_1, x_2 \geq 0. \end{aligned}$$

Because systems of equations are a familiar setting from linear algebra, we introduce additional variables, called *slack variables*, corresponding to these three inequalities. Labeling these variables as s_1, s_2, and s_3, we then have

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 \\ &\text{subject to} \\ &x_1 + s_1 = 8 \\ &2x_1 + 2x_2 + s_2 = 28 \\ &3x_1 + 2x_2 + s_3 = 32 \\ &x_1, x_2, s_1, s_2, s_3 \geq 0. \end{aligned} \tag{1.8}$$

We will use the terminology that (1.8) is the original LP (1.6) expressed in *standard form*.

Note that requiring the slack variables to be nonnegative assures us that our three equations are equivalent to the three inequalities from (1.6). Recalling the matrix inequality form notation (1.4) and denoting the vector of slack variables by

$$\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix},$$

we can express the *standard matrix form* as follows:

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &[A|I_3] \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{b} \\ &\mathbf{x}, \mathbf{s} \geq \mathbf{0}, \end{aligned} \tag{1.9}$$

where I_3 denotes the 3-by-3 identity matrix. Thus we have converted the $m = 3$ inequalities in $n = 2$ decision variables from the original LP into a matrix equation, or system of equations, involving $m = 3$ equations in $m + n = 5$ unknowns. Note in particular that $[A|I_3]$ is a 3 by 5 matrix and that $\begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix}$ belongs to $\mathbb{R}^5$.

It is important to recognize connections between solutions to this system and the problem at hand.

Waypoint 1.3.1. Before proceeding, explain why the matrix equation $[A|I_3]\begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{b}$ has infinitely many solutions.

We now seek to determine the set of solutions to $[A|I_3]\begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{b}$, ignoring momentarily the sign restrictions placed on the variables in the LP. Recall that the *row space*, resp. *column space* of a matrix is the set of all linear combinations of the row vectors (resp. column vectors) that comprise the matrix. The dimensions of the row and column spaces are equal, having common value referred to as the *rank* of the matrix. In the *FuelPro* example,

$$[A|I_3] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix},$$

which, as elementary row operations demonstrate, has pivots in three columns. Therefore $[A|I_3]$ has rank $m = 3$.

On the other hand, the *null space* of $[A|I_3]$ is the set of all solutions to the homogeneous matrix equation $[A|I_3]\,\mathbf{v} = \mathbf{0}$. By the Rank-Nullity Theorem (Theorem B.6.1), the sum of its dimension, referred to as the *nullity*, and the rank equals the number of columns of $[A|I_3]$, which is $m + n = 5$. Hence, the dimension of the null space is $n = 2$.

Finally, we recall that the solution to a consistent matrix equation $[A|I_3]\begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{b}$ can be written in the form

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{v}_h + \mathbf{v}_p,$$

where $\mathbf{v}_h$ is the general solution to the homogeneous matrix equation $[A|I_3]\,\mathbf{v} =$

$\mathbf{0}$ and $\mathbf{v}_p$ is a particular solution to the nonhomogeneous equation $[A|I_3]\,\mathbf{v} = \mathbf{b}$. Therefore if the null space of $[A|I_3]$ is dimension $n = 2$, we can expect $[A|I_3]\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix} = \mathbf{b}$ to have $n = 2$ free variables in its general solution, provided the matrix equation is consistent.

◆ ——— ◆

Waypoint 1.3.2. For the *FuelPro* LP, determine the general solution to the matrix equation $[A|I_3]\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix} = \mathbf{b}$. For each extreme point in Figure 1.5, choose the free variables in a way that produces the given extreme point. Next to the extreme point, label the vector $\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix}$ with all <u>five</u> of its entries.

◆ ——— ◆

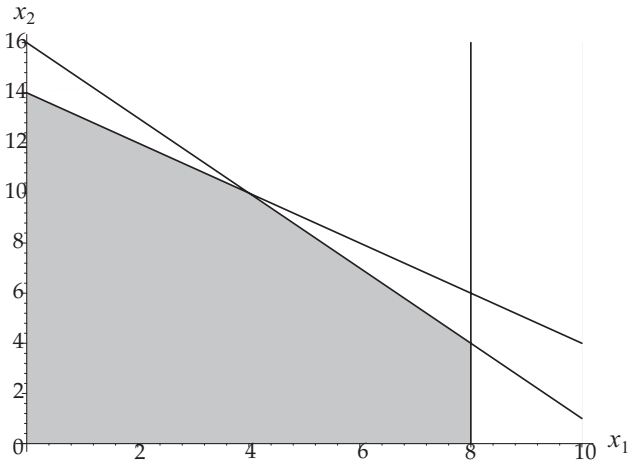


FIGURE 1.5: Feasible region for *FuelPro* LP.

The solutions to the preceding matrix equation play a special role in the context of the LP, in that they correspond to what we call *basic solutions*.

Definition 1.3.1. Consider an LP consisting of m inequalities and n decision variables, whose matrix inequality form is given by (1.4). Then the constraints of the *standard matrix form* of this LP are represented by the matrix equation,

$$[A|I_m]\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix} = \mathbf{b}, \tag{1.10}$$

where, $\mathbf{x}$ is an n by 1 vector of decision variables and $\mathbf{s}$ is an m by 1 vector of slack variables. A basic solution to the LP is formed by first setting to zero n variables of $\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix}$. These variables are referred to as the *nonbasic variables*. If the m columns corresponding to the remaining variables form a linearly independent set, then there exist unique values for the remaining variables, which we call *basic variables*. Taken together, these nonbasic and basic variables constitute a basic solution of LP (1.4).

The fact that the basic variables, denoted by the vector $\mathbf{x}'$, are uniquely determined in Definition 1.3.1 follows from the Invertible Matrix Theorem and the assumption that the m columns form a linearly independent set. For when this occurs, these columns yield an m by m invertible submatrix, B, of $[A|I_m]$, so that $B\mathbf{x}' = \mathbf{b}$ has a unique solution.

Here is the connection to commonly used terminology in linear algebra. When solving a system of equations having infinitely many solutions, we obtain two types of variables, those considered free and those considered basic. While there exist different possible ways to assign which variables are free, their number must equal the nullity of the matrix. The values of the free variables uniquely determine the basic variable values. In the terminology of Definition 1.3.1, nonbasic variables can be viewed as free variables all of which have been set to zero.

Basic solutions correspond to intersection points of the constraints of the original LP. Unfortunately, this definition is not precise in that it can also include points that lie outside the LP's feasible region. In particular, observe that in the preceding Waypoint, intersection points outside the feasible region corresponded to vectors $\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix}$, in which at least one entry is negative. We therefore refine our definition of basic solution as follows.

Definition 1.3.2. A basic feasible solution (BFS) to an LP is a basic solution in which all basic variables are nonnegative.

For the *FuelPro* example, basic feasible solutions correspond to the points $(0,0)$, $(8,0)$, $(8,4)$, $(4,10)$, and $(0,14)$.

Waypoint 1.3.3. Consider the minimization problem

$$\begin{aligned} &\text{minimize } z = -5x_1 + 2x_2 \\ &\text{subject to} \\ &\quad x_1 \leq 2 \\ &3x_1 + 2x_2 \leq 6 \\ &8x_1 + 3x_2 \leq 12 \\ &\quad x_1, x_2 \geq 0. \end{aligned}$$

1. Sketch the feasible region for the LP. Label all points of intersection of the constraints.
2. Introduce slack variables and express the LP in standard matrix form.
3. Determine the basic solutions and basic feasible solutions for the LP. Complete this last task without actually performing elementary row operations. Instead use the points of intersection determined in (1).
4. Graphically solve the LP by using contours of the objective function.

Exercises Section 1.3

1. For each of the following LPs, write the LP in the standard matrix form

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &[A|I_m]\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix} = \mathbf{b} \\ &\quad \mathbf{x}, \mathbf{s} \geq \mathbf{0}. \end{aligned}$$

Then determine all basic solutions, expressing each solution in vector form and indicating whether it is also basic feasible. Label each basic feasible solution next to its corresponding point on the feasible region graph.

(a)

$$\begin{aligned} \text{maximize } z &= 6x_1 + 4x_2 \\ \text{subject to} & \\ 2x_1 + 3x_2 &\leq 9 \\ x_1 &\geq 4 \\ x_2 &\leq 6 \\ x_1, x_2 &\geq 0 \end{aligned}$$

(b)

$$\begin{aligned} \text{minimize } z &= -x_1 + 4x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\geq 5 \\ x_1 + x_2 &\geq 4 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

2. Show that in the *FuelPro* LP that no basic solution exists for which x_1 and s_1 are nonbasic. This result underscores the importance of the linear independence condition in Definition 1.3.1.

3. The following LP has its feasible region given by the segment connecting the points $\mathbf{x} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$:

$$\begin{aligned} \text{maximize } z &= 3x_1 + 2x_2 \\ \text{subject to} & \\ x_1 &\leq 4 \\ x_1 + 3x_2 &\leq 15 \\ 2x_1 + x_2 &= 10 \\ x_1, x_2 &\geq 0. \end{aligned}$$

   (a) By rewriting the third constraint as the combination of two others, express the LP in matrix inequality form. Then express it in standard matrix form.

   (b) The standard matrix form has four slack variables, two of which correspond to the third constraint. Show that when these two are chosen as nonbasic, the resulting system of equations in the basic variables infinitely many solutions, which correspond to the segment connecting $\mathbf{x} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$.

4. According to Definition 1.3.2, a basic feasible solution is a basic solution in which all basic variables are nonnegative. An LP is said to be *degenerate* if it has a basic feasible solution in which at least one basic variable equals zero. Experiment and construct an example of a degenerate LP. (Hint: Start with the *FuelPro* LP and add a constraint, whose boundary line intersects the original LP at one of its basic feasible solutions.)

5. A subset V of $\mathbb{R}^n$ is said to be *convex* provided whenever two points belong to V, so does the line segment connecting them. In other words, $\mathbf{x}_1, \mathbf{x}_2 \in V$, implies that $t\mathbf{x}_1 + (1-t)\mathbf{x}_2 \in V$ for all $0 \leq t \leq 1$.

   (a) Use the matrix inequality form of an LP to show that the feasible region of an LP is convex.

   (b) Show that if $\mathbf{x}_1$ and $\mathbf{x}_2$ are solutions of an LP, so is $t\mathbf{x}_1 + (1-t)\mathbf{x}_2$ for any $0 \leq t \leq 1$.

6. Given a set of vectors $V = \left\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\right\}$ in $\mathbb{R}^n$, a *convex combination* of the vectors is any linear combination of the form

$$\sum_{i=1}^{p} \sigma_i \mathbf{x}_i = \sigma_1 \mathbf{x}_1 + \sigma_2 \mathbf{x}_2 + \cdots + \sigma_p \mathbf{x}_p,$$

   where the weights σ_i satisfy $\sigma_i \geq 0$ for all i and where $\displaystyle\sum_{i=1}^{p} \sigma_i = 1$. The set of all convex combinations of vectors in V is referred to as the *convex polyhedron*, or *convex hull*, generated by V. Suppose that the feasible region of the LP is the convex polyhedron generated by its basic feasible solutions, $V = \left\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\right\}$. Show that if the LP has an optimal solution, then it has a basic feasible solution that is also optimal. (Hint: Start with an LP having optimal solution $\tilde{\mathbf{x}}$. Express $\tilde{\mathbf{x}}$ as a convex combination of the basic feasible solutions and show that the objective function evaluated at one of the basic feasible solutions is at least as large as the objective function evaluated at $\tilde{\mathbf{x}}$.)

Chapter 2

The Simplex Algorithm

2.1 The Simplex Algorithm

Considered to be the classical approach for solving LPs, the simplex algorithm was developed in 1947 by George Dantzig, who used it to solve "programming" problems for the Air Force. Such logistics problems addressed a variety of issues including, for example, the scheduling of troop deployments and the delivery of supplies.

2.1.1 An Overview of the Algorithm

We begin by revisiting the standard form of the *FuelPro* LP:

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 \\ &\text{subject to} \\ x_1 \quad &+ s_1 \quad = 8 \\ 2x_1 + 2x_2 \quad &+ s_2 \quad = 28 \\ 3x_1 + 2x_2 \quad &+ s_3 = 32 \\ x_1, x_2, s_1, s_2, s_3 &\geq 0. \end{aligned} \tag{2.1}$$

We will rewrite the LP as an "augmented matrix," so to speak. That is, we will write the system of equations given by the constraints as an augmented matrix, and we will add a top row that records the value of our objective function. The result is shown in Table 2.1.

TABLE 2.1: Initial tableau for *FuelPro* Petroleum problem

z	x_1	x_2	s_1	s_2	s_3	RHS
1	-4	-3	0	0	0	0
0	1	0	1	0	0	8
0	2	2	0	1	0	28
0	3	2	0	0	1	32

Sometimes Table 2.1 is referred to as the "simplex tableau."

In a nutshell, the simplex algorithm works as follows:

1. We begin by finding an initial basic feasible solution (BFS). For many of the initial problems we shall encounter, this task is readily accomplished by setting all decision variables equal to zero.

2. We use the top row of the tableau to determine which nonbasic variable should then become positive in order to increase the objective function most rapidly. Because this variable will switch from nonbasic to basic, we call it the *entering variable*.

3. We then find the basic variable that swaps places with the entering variable and becomes nonbasic.

4. Once we determine which variables trade roles, we perform elementary row operations, whose purpose is to yield a new tableau containing updated values for all variables, as well as the objective function.

5. We repeat the previous steps until the objective function can no longer be increased. The basic feasible solution at that stage is the optimal solution.

2.1.2 A Step-By-Step Analysis of the Process

We now break down the algorithm into greater detail by using it to solve the LP 2.1.

1. We start by finding an initial BFS. In the *FuelPro* example, this is easily accomplished by setting $x_1 = x_2 = 0$, in which case the slack variables are given by $s_1 = 8$, $s_2 = 28$, and $s_3 = 32$. This BFS corresponds to the origin in the feasible region. We have BV $= \{s_1, s_2, s_3\}$ and NBV $= \{x_1, x_2\}$. (Note, we do not consider z as being either BV or NBV in this process.) Later, we will discuss how to handle situations when the origin does not correspond to a BFS.

2. Our objective function is given by $z = f(x_1, x_2) = 4x_1 + 3x_2$. We see that increasing the nonbasic variable x_1 by a set amount will increase z more than will increasing the other nonbasic variable x_2 by the same amount. We thus choose x_1 as the entering variable and focus on its corresponding column. Note that in general, selecting the entering variable in a maximization problem comes down to determining the nonbasic variable whose coefficient in the top row is most negative.

3. We now determine the extent to which x_1 may increase by focusing on

the constraint equation corresponding to each row. Each such equation involves x_1 and a basic variable. We have the following three equations:

$$x_1 + s_1 = 8$$
$$2x_1 + s_2 = 28$$
$$3x_1 + s_3 = 32.$$

Currently, $s_1 = 8, s_2 = 28$, and $s_3 = 32$. Since, the values of these variables must remain nonnegative, x_1 can only increase by so much before at least one of these values becomes zero. In general, each equation allows x_1 to increase up to a value of

(RHS in Equation)/(Coeff of x_1 in Equation).

In this case, the three equations permit x_1 to increase up to values of 8, 14, and $\frac{32}{3}$, respectively. Since 8 is the smallest of these three values, x_1 will replace s_1 as a basic variable. We sometimes call this process of computing ratios to determine which variable becomes nonbasic, performing the *ratio test*.

4. We now pivot on the entry in the column corresponding to x_1 and the row corresponding to s_1. This entry is highlighted in Table 2.1. The purpose of this pivot is to update values of all variables and the objective function when x_1 and s_1 switch roles. The result is the new tableau in Table 2.2.

TABLE 2.2: Tableau after first iteration

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	-3	4	0	0	32
0	1	0	1	0	0	8
0	0	2	-2	1	0	12
0	0	2	-3	0	1	8

We have now completed one iteration of the simplex algorithm. The new BFS corresponds to the extreme point at $(8, 0)$. At this stage, BV $= \{x_1 = 8, s_2 = 12, s_3 = 8\}$, NBV $= \{x_2, s_1\}$, and $z = 32$.

5. We now repeat this process. The only negative nonbasic variable coefficient in the top row of Table 2.2 corresponds to x_2, so x_2 becomes the entering variable. Since x_1 is now basic, the constraints are given by

$$x_1 + 0x_2 = 8$$
$$2x_2 + s_2 = 12$$
$$2x_2 + s_3 = 8.$$

We again apply the ratio test. Note that the first equation places no restriction on the growth of x_2. The remaining two equations limit the increase in x_2 to 6 and 4, respectively. We thus focus on the third equation, which tells us that x_2 replaces s_3 as a basic variable, and we pivot on the entry highlighted in Table 2.2. The new tableau is given in Table 2.3.

TABLE 2.3: Tableau after second iteration

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	$-\frac{1}{2}$	0	$\frac{3}{2}$	44
0	1	0	1	0	0	8
0	0	0	1	1	-1	4
0	0	1	$-\frac{3}{2}$	0	$\frac{1}{2}$	4

The new BFS corresponds to the extreme point at $(8, 4)$. We have BV $= \{x_1 = 8, x_2 = 4, s_2 = 4\}$, NBV $= \{s_1, s_3\}$ and $z = 44$.

6. The process is repeated one last time. The variable s_1 becomes basic. The corresponding ratios are given by 8, 4, and $-\frac{8}{3}$. Since the negative ratio corresponds to unlimited growth in s_1, the value of 4 "wins" the ratio test. Thus s_1 replaces s_2 as a basic variable and we pivot on the highlighted entry in Table 2.3. The resulting tableau appears in Table 2.4.

TABLE 2.4: Tableau after third iteration

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	0	$\frac{1}{2}$	1	46
0	1	0	0	-1	1	4
0	0	0	1	1	-1	4
0	0	1	0	$\frac{3}{2}$	-1	10

The new BFS corresponds to the extreme point at $(4, 10)$. We have BV $= \{x_1 = 4, x_2 = 10, s_1 = 4\}$, NBV $= \{s_2, s_3\}$ and $z = 46$. Note that all coefficients in the top row of the tableau are nonnegative, so we have obtained the optimal solution, and *FuelPro* maximizes its profit if $x_1 = 4, x_2 = 10$, in which case $z = 46$, i.e., a profit of \$4.60 per hour. These results agree with those obtained using Maple's `LPSolve` command and graphical tools from Section 1.2.

Here are points to keep in mind regarding this process.

- Performing elementary row operations is a tedious task. Consequently, one's focus should be on global aspects of the algorithm such as keeping

track of the basic and nonbasic variables and understanding the rationale behind the ratio test. Doing so will prove beneficial during later discussions of special cases, duality, and sensitivity analysis. A Maple worksheet found at the end of this section can facilitate this process.

- The preceding example was fairly straightforward. As such, it raises a variety of questions. For example, "What happens in the process if the LP is unbounded? What happens if the LP has alternative optimal solutions? What happens if the LP has equality constraints?" We address all these questions in the next section.

2.1.3 Solving Minimization Problems

The simplex algorithm is easily modified to solve minimization problems in which $\mathbf{x} = \mathbf{0}$ corresponds to a basic feasible solution. Suppose, for example, that we consider the following minimization LP from Section 1.3:

$$\begin{aligned} \text{minimize } z &= -5x_1 + 2x_2 \\ \text{subject to} & \\ x_1 &\leq 2 \\ 3x_1 + 2x_2 &\leq 6 \\ 8x_1 + 3x_2 &\leq 12 \\ x_1, x_2 &\geq 0. \end{aligned} \tag{2.2}$$

There are two different approaches to solving (2.2):

1. Rephrase the problem as a maximization problem. That is, keep each constraint above as is, but replace the objective with

$$\text{maximize } \tilde{z} = -(-5x_1 + 2x_2) = 5x_1 - 2x_2.$$

   Now solve the LP using the algorithm exactly as before.

2. Maintain the objective goal of minimizing z and modify the simplex algorithm slightly as follows. At each stage, instead of choosing the entering variable on the basis of the most negative coefficient in the top row of the tableau, select the entering variable on the basis of the most positive coefficient. (Think about why this works!) Then terminate the algorithm when entries in the top row of the tableau corresponding to nonbasic variables are all nonpositive.

Either of these methods works; our convention will be to use the latter. For example, let us use this approach to solve LP (2.2). The initial tableau is given in Table 2.5, in which case BV $= \{s_1 = 2, s_2 = 6, s_3 = 12\}$, and NBV $= \{x_1, x_2\}$.

Since 5 is the most positive nonbasic variable coefficient in the top row, x_1 is

TABLE 2.5: Initial tableau for minimization LP (2.2)

z	x_1	x_2	s_1	s_2	s_3	RHS
1	5	-2	0	0	0	0
0	1	0	1	0	0	2
0	3	2	0	1	0	6
0	$\boxed{8}$	3	0	0	1	12

the entering variable. The third constraint row yields the smallest ratio of $\frac{3}{2}$, so s_3 becomes nonbasic and is replaced by x_1. The new tableau is shown in Table 2.6.

TABLE 2.6: Tableau after first iteration for minimization LP (2.2)

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	$-\frac{31}{8}$	0	0	$-\frac{5}{8}$	$-\frac{15}{2}$
0	0	$-\frac{3}{8}$	1	0	$-\frac{1}{8}$	$\frac{1}{2}$
0	0	$\frac{7}{8}$	0	1	$-\frac{3}{8}$	$\frac{3}{2}$
0	1	$\frac{3}{8}$	0	0	$\frac{1}{8}$	$\frac{3}{2}$

The new BFS corresponds to the extreme point at $\left(\frac{3}{2}, 0\right)$, where BV $= \left\{x_1 = \frac{3}{2}, s_1 = \frac{1}{2}, s_2 = \frac{3}{2}\right\}$, NBV $= \{x_2, s_3\}$ and $z = -\frac{15}{2}$. Since all nonbasic variable coefficients in the top row of the tableau are nonpositive, this solution is optimal.

Waypoint 2.1.1. Now solve the minimization LP (2.2) by converting it to a maximization problem.

2.1.4 A Step-by-Step Maple Implementation of the Simplex Algorithm

What follows is a worksheet, entitled **Simplex Algorithm.mw**, which facilitates using the simplex algorithm to solve an LP stated in matrix inequality

form as

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} \leq \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{2.3}$$

In this case, the worksheet is implemented to solve the *FuelPro* LP. The user enters the quantities $\mathbf{c}$, A, and $\mathbf{b}$, as well as the number of constraints and number of decision variables. (The `Vector` command is used to enter $\mathbf{c}$ as a row vector.) A `for-do` loop structure then creates arrays of decision variables and slack variables of appropriate respective sizes. These quantities are used to create a matrix labeled `LPMatrix`. Note that the task of keeping track of basic versus nonbasic variables is still left to the user.

Three Maple *procedures* are then created to facilitate implementation of the simplex algorithm. A procedure is a small "program" within a worksheet that accepts certain arguments and performs specified tasks. The first procedure, `Tableau`, merely prints the tableau matrix with a top row of variable labels. The second procedure, `RowRatios(M,c)` performs the ratio test on a matrix `M` using column number `c`, where column 0 corresponds to the leftmost column of the tableau. Once this procedure determines the ratio test results, the third procedure, `Iterate(M,c,r)`, pivots on the entry (`c`,`r`) of the matrix `M`, where column 0 corresponds to the leftmost column and row 0 to the top row of the tableau, not including the variable labels. The term `local` within each procedure refers to one or more variables that are used locally to the procedure only and whose values are not accessible outside of the procedure.

```
> restart;with(LinearAlgebra):
> c:=Vector[row]([4,3]);
  # Create row vector with objective coefficients.
```

$$c := [4, 3]$$

```
> A:=Matrix(3,2,[1,0,2,2,3,2]);
  # Matrix of constraint coefficients.
```

$$A := \begin{bmatrix} 1 & 0 \\ 2 & 2 \\ 3 & 2 \end{bmatrix}$$

```
> b:=<8,28,32>;
  # Constraint bounds.
```

$$b := \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}$$

```
> n:=2:m:=3:
  # Enter the number of decision variables and constraints, respectively.
> x:=array(1..n):s:=array(1..m):
  # Create arrays of decision and slack variables.
> Labels:=Matrix(1,2+n+m,[z,seq(x[i],i=1..n),seq(s[j],j=1..m),RHS]):
  # Create a top row of variable labels for the tableau
> LPMatrix:=<UnitVector(1,m+1) | <-c,A>  | <ZeroVector[row](m),
  IdentityMatrix(m)>  | <0,b>>;
  # Create matrix corresponding to the tableau.
```

$$LPMatrix := \begin{bmatrix} 1 & -4 & -3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 2 & 2 & 0 & 1 & 0 & 28 \\ 0 & 3 & 2 & 0 & 0 & 1 & 32 \end{bmatrix}$$

```
> Tableau:=proc(M);return(<labels,M>):end:
  # Procedure for printing tableau with labels.
> RowRatios:=proc(M,c) local k:
  for k from 2 to nops(convert(Column(M,c+1),list)) do
  if M[k,c+1]=0 then print(cat(``Row '', convert(k-1,string), ``
  Undefined''))
  else print(cat(``Row '',convert(k-1,string), ``Ratio = '',
  convert(evalf(M[k,nops(convert(Row(M,k),list))]/M[k,c+1]),string)))
  end if; end do;end:
  #The ratio test procedure applied to column c of M.
> Iterate:=proc(M,r,c) RowOperation(M,r+1,(M[r+1,c+1])^(-1),inplace=true):
  Pivot(M,r+1,c+1,inplace=true):return(Tableau(M)):end:
  # Iteration of the simplex algorithm applied to row r, column
  c
> Tableau(LPMatrix);
  # Display initial tableau.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & -4 & -3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 2 & 2 & 0 & 1 & 0 & 28 \\ 0 & 3 & 2 & 0 & 0 & 1 & 32 \end{bmatrix}$$

```
> RowRatios(LPMatrix,1);
  # Determine ratios corresponding to column 1.
```

"Row 1 Ratio = 8."

"Row 2 Ratio = 14."

"Row 3 Ratio = 10.66666667."

```
> Iterate(LPMatrix,1,1);
  # Pivot on entry in row 1, column 1 of matrix.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & 0 & -3 & 4 & 0 & 0 & 32 \\ 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 0 & 2 & -2 & 1 & 0 & 12 \\ 0 & 0 & 2 & -3 & 0 & 1 & 8 \end{bmatrix}$$

```
> RowRatios(LPMatrix,2);
  # Determine ratios corresponding to column 2.
```

"Row 1 Ratio Undefined."

"Row 2 Ratio = 6."

"Row 3 Ratio = 4."

```
> Iterate(LPMatrix,3,2);
  # Pivot on entry in row 3, column 2 of matrix.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & 0 & 0 & -\frac{1}{2} & 0 & \frac{3}{2} & 44 \\ 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 & 1 & -1 & 4 \\ 0 & 0 & 1 & -\frac{3}{2} & 0 & \frac{1}{2} & 4 \end{bmatrix}$$

```
> RowRatios(LPMatrix,3);
  # Determine ratios corresponding to column 3.
```

"Row 1 Ratio =8."

"Row 2 Ratio = 4."

"Row 3 Ratio = -2.666666667."

```
> Iterate(LPMatrix,2,3);
  # Pivot on entry in row 2, column 3 of matrix to obtain final
  tableau.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & 0 & 0 & 0 & \frac{1}{2} & 1 & 46 \\ 0 & 1 & 0 & 0 & -1 & 1 & 4 \\ 0 & 0 & 0 & 1 & 1 & -1 & 4 \\ 0 & 0 & 1 & 0 & \frac{3}{2} & -1 & 10 \end{bmatrix}$$

Exercises Section 2.1

1. Use the simplex algorithm to solve each of the following LPs.

   (a)

$$\begin{aligned} &\text{maximize } z = 3x_1 + 2x_2 \\ &\text{subject to} \\ &x_1 \leq 4 \\ &x_1 + 3x_2 \leq 15 \\ &2x_1 + x_2 \leq 10 \\ &x_1, x_2 \geq 0 \end{aligned}$$

   (b)

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 \\ &\text{subject to} \\ &x_1 \leq 4 \\ &-2x_1 + x_2 \leq 12 \\ &x_1 + 2x_2 \leq 14 \\ &x_1, x_2 \geq 0 \end{aligned}$$

   (c)

$$\begin{aligned} &\text{maximize } z = 4x_1 + x_2 + 5x_3 \\ &\text{subject to} \\ &2x_1 + x_2 + 3x_3 \leq 14 \\ &6x_1 + 3x_2 + 3x_3 \leq 22 \\ &2x_1 + 3x_2 \leq 14 \\ &x_1, x_2, x_3 \geq 0 \end{aligned}$$

(d)

$$\begin{aligned} \text{minimize } z &= 3x_1 - 2x_2 \\ \text{subject to} & \\ x_1 - 2x_2 &\leq 2 \\ x_1 + x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

2. Try to use the simplex algorithm to solve the minimization LP

$$\begin{aligned} \text{minimize } z &= -x_1 + 4x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\geq 8 \\ x_1 + x_2 &\geq 4 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

   What happens? Specify the attribute of this problem that makes solving it more challenging than solving the minimization problem in Exercise 1.

3. A constraint is said to be *binding* at the optimal solution of an LP if there is equality in that constraint in the optimal solution. For the first two LPs in Exercise 1, determine which constraints are binding.

2.2 Alternative Optimal/Unbounded Solutions and Degeneracy

When applying the simplex algorithm, how can we determine from the tableau whether an LP has alternative optimal solutions or is unbounded? The purpose of this section is to address these special cases.

2.2.1 Alternative Optimal Solutions

Consider first the LP given by the following:

$$\begin{aligned} &\text{maximize } z = 12x_1 + 3x_2 \\ &\text{subject to} \\ &\quad 4x_1 + x_2 \leq 8 \\ &\quad 5x_1 + 2x_2 \leq 12 \\ &\quad x_1, x_2 \geq 0. \end{aligned} \tag{2.4}$$

The feasible region for the LP is shown in Figure 2.1. Superimposed on the feasible region is the contour $z = 12$ of the objective function. The fact that the contour $z = 12$ and all others are parallel to one edge of the feasible region can be immediately seen by observing that the objective function is a scalar multiple of the left-hand side of the first constraint.

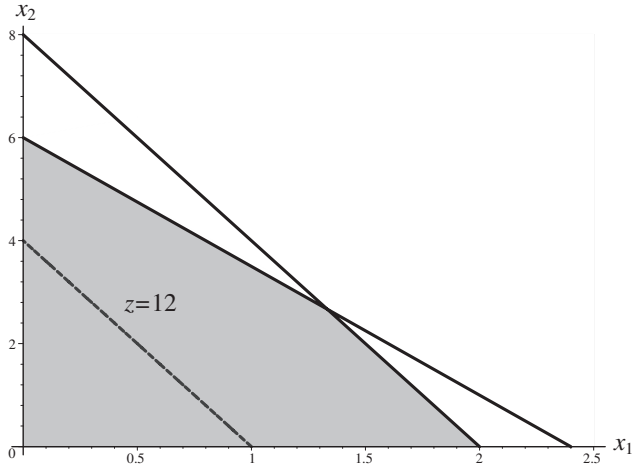


FIGURE 2.1: Feasible region for LP 2.4.

After introducing two slack variables s_1 and s_2 and performing one iteration of the simplex algorithm, we arrive at the tableau given in Table 2.7.

TABLE 2.7: LP (2.4) after one iteration

z	x_1	x_2	s_1	s_2	RHS
1	0	0	3	0	24
0	1	$\frac{1}{4}$	$\frac{1}{4}$	0	2
0	0	$\frac{3}{4}$	$-\frac{5}{4}$	1	2

We have BV $= \{x_1 = 2, s_2 = 2\}$, which corresponds to the extreme point at $\mathbf{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$.

The nonbasic variables are given by NBV $= \{x_2, s_1\}$ and neither has a negative coefficient in the top row, thereby indicating that the current solution is optimal. But let's take a closer look. The top row of the tableau dictates that increasing s_1 will decrease the value of z. (In fact, it will return us to the initial basic feasible solution at the origin!) On the other hand, *increasing the other nonbasic variable x_2 from its current value of zero will not change the current z value because x_2 currently has a coefficient of zero in the top row*. An additional iteration of the simplex algorithm in which x_2 becomes basic leads to $x_1 = \frac{4}{3}$, $x_2 = \frac{8}{3}$, and $s_1 = s_2 = 0$. Thus, each of the vectors $\mathbf{x} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} \frac{4}{3} \\ \frac{8}{3} \end{bmatrix}$ are solutions to the LP. By exercise 5, of Section 1.3, so is every point on the line segment connecting the two vectors:

$$\mathbf{x} = (1-t)\begin{bmatrix} 2 \\ 0 \\ 0 \\ 2 \end{bmatrix} + t\begin{bmatrix} \frac{4}{3} \\ \frac{8}{3} \\ 0 \\ 0 \end{bmatrix},$$

where $t \in [0, 1]$. This infinite set of vectors corresponds to points along the edge of feasible region connecting $(2, 0)$ and $\left(\frac{4}{3}, \frac{8}{3}\right)$.

2.2.2 Unbounded Solutions

How do we determine whether an LP is unbounded from its tableau? (Recall that for a maximization problem, an unbounded LP is one in which the objective function can be made arbitrarily large.) To answer this question, we consider an LP whose tableau at some stage of the simplex algorithm is almost identical to that from Table 2.2 of Section 2.1, such as the one shown in Table 2.8.

.

TABLE 2.8: Example tableau similar to 2.2

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	-3	4	0	0	32
0	1	0	1	0	0	8
0	0	■	-2	1	0	12
0	0	■	-3	0	1	8

Assume that x_1, s_2, and s_3 are basic. The variable x_2 is the entering variable, and clearly any strict increase in its value will lead to a strict increase in z. The row of the tableau corresponding to the first constraint places no limit on the increase of x_2 due to the coefficient of zero in the x_2 column. Thus, if either of the entries marked ■ were also zero, then the corresponding constraints would also place no limit on the increase of x_2. But the same conclusion can be made if either of the entries marked ■ were less than zero. The variable x_2 could increase as large as desired, and slack variables could increase in a positive manner ensuring that all constraint equations were still satisfied. *Thus, if a variable in the top row of an LP maximum problem has a negative coefficient and all the entries below the top row are less than or equal to zero, then the LP is unbounded.*

Waypoint 2.2.1. The tableau shown in Table 2.9 results when the simplex algorithm is used to solve a standard maximization LP. Assume that $a \geq 0$ and that BV $= \{x_1, s_1\}$.

1. For which values of a does the LP have multiple optimal solutions?
2. For which values of a is the LP unbounded?
3. For which values of a is the LP degenerate?

TABLE 2.9: Tableau for a standard maximization problem

z	x_1	x_2	s_1	s_2	RHS
1	0	$1-a$	0	$2-a$	1
0	1	0	0	-1	a
0	0	$a-3$	1	$a-4$	2

2.2.3 Degeneracy

Recall that Definition 1.3.2 states the basic variables in any basic feasible solution must be nonnegative. For the examples we have considered thus far, basic variables have always been positive. A *degenerate* LP is one that possesses a basic feasible solution in which at least one basic variable equals zero. Degenerate LPs are important from the standpoint of convergence of the simplex algorithm.

An important observation regarding the simplex algorithm is that from one iteration to the next, the objective function increases by an amount equal to the value of the entering variable multiplied by the negative of that variable's coefficient in the top row of the tableau. Thus, if an LP is nondegenerate, then the new value of each entering variable must be nonzero so that the objective function must strictly increase at each iteration. In particular, the simplex algorithm cannot encounter the same basic feasible solution, so the algorithm must terminate due to the finite number of such solutions.

For a degenerate LP, the objective function can remain unchanged over successive iterations. One of the simplest examples illustrating this possibility is given by the LP

$$\begin{aligned} &\text{maximize } z = 5x_1 + 2x_2 \\ &\quad \text{subject to} \\ &\qquad x_1 + x_2 \le 6 \\ &\qquad x_1 - x_2 \le 0 \\ &\qquad x_1, x_2 \ge 0, \end{aligned} \tag{2.5}$$

whose feasible region is shown in Figure 2.2.

The initial tableau for the LP is given in Table 2.10.

TABLE 2.10: Initial tableau for 2.5

z	x_1	x_2	s_1	s_2	RHS
1	-5	-2	0	0	0
0	1	1	1	0	6
0	1	-1	0	1	0

If we start with our usual initial basic feasible solution $x_1 = x_2 = 0$, we see that $s_2 = 0$, so this LP is degenerate. From a graphical perspective, if we temporarily view the sign conditions as constraints, then we can see that the LP is degenerate by observing there is an extreme point where equality holds for three, as opposed to just two, of the four constraints.

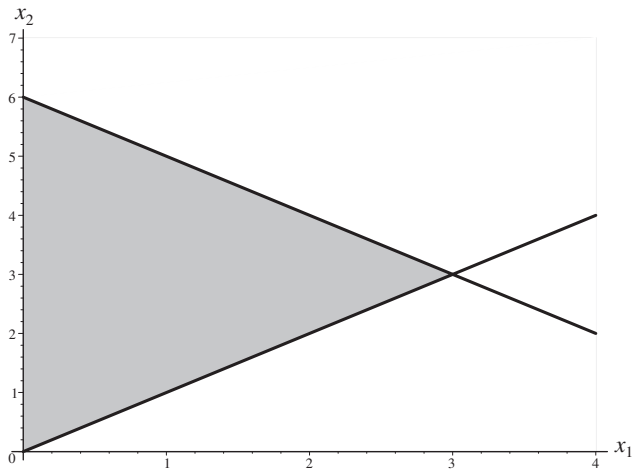


FIGURE 2.2: Feasible region for a degenerate LP 2.5.

Following our "usual rules," we allow x_1 to become basic, but we observe that, due to degeneracy, the ratio corresponding to the bottom row is zero. To determine which of s_1 or s_2 becomes nonbasic, we consider the equations corresponding to rows 1 and 2. Since x_2 remains nonbasic, the equation in row 1 corresponds to $x_2 + s_1 = 6$, whereas that in row 2 yields $x_1 + s_2 = 0$. The first equation permits x_1 to increase by 6. However, in the second equation, the basic variable $s_2 = 0$ so that x_1 cannot increase without s_2 becoming negative. If we treat the ratio in the bottom row as the smallest "positive" ratio and pivot in a way that x_1 replaces s_2 as a nonbasic variable, we obtain the new tableau in Table 2.11.

TABLE 2.11: Tableau after first iteration for 2.5

z	x_1	x_2	s_1	s_2	RHS
1	0	-7	0	5	0
0	0	2	1	-1	6
0	1	-1	0	1	0

Note that the objective function value has not changed from zero nor has the extreme point in the feasible region. Only have the roles of x_1 and s_2 as basic versus nonbasic changed; their values after the first iteration remain equal to zero.

At the next iteration, x_2 becomes basic, and we are once again faced with a ratio of 0. In this case, the equations corresponding to the bottom two rows are given by $2x_2 + s_1 = 6$ and $x_1 - x_2 = 0$. The first permits x_1 to increase by

3. The second permits x_2 to increase without bound, provided x_1 does so as well in a manner that their difference remains equal to 0. Thus, we now let x_2 replace s_1 as a basic variable, and the final tableau becomes that shown in Table 2.12

TABLE 2.12: Tableau after second iteration for 2.5

z	x_1	x_2	s_1	s_2	RHS
1	0	0	$\frac{7}{2}$	$\frac{3}{2}$	21
0	0	1	$\frac{1}{2}$	$-\frac{1}{2}$	3
0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	3

We see that the optimal solution corresponds to the extreme point, $\begin{bmatrix} 3 \\ 3 \end{bmatrix}$ with an objective value of $z = 21$.

In this example, degeneracy led to ratios of zero, a situation we have not encountered thus far when performing the ratio test. But degeneracy can also result in ties when we perform the ratio test, implying there exists more than one choice for the departing basic variable. Endless cycling between basic feasible solutions is possible depending upon how such ties are resolved. In real-life applications of linear programming, this problem rarely occurs. In fact, any degenerate LP in which cycling occurs must contain at least six decision variables [30]. Moreover, *Bland's Rule* is an advanced linear programming tool that overcomes the theoretical difficulty of cycling. Simply put, if the variables of the LP are listed as a single variable with an increasing subscript, it dictates a "smallest subscript rule." At any stage of the simplex algorithm, given a choice of two variables to enter the set of basic variables or to leave the set of basic variables, always choose the variable with the smaller subscript.

Exercises Section 2.2

1. Use the simplex algorithm to verify that the LP given by

$$\begin{aligned} \text{minimize } z &= 2x_1 + 6x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\leq 6 \\ x_2 &\leq 1 \\ x_1, x_2 &\geq 0, \end{aligned}$$

has alternative optimal solutions.

2. State a rule that can be used to determine from a tableau whether an LP minimization problem is unbounded.

3. The tableau given in (2.13) corresponds to that obtained in the process of using the simplex algorithm to solve an LP whose objective is to maximize a function of x_1 and x_2. Assume that the variables s_1 and s_2 are nonbasic.

TABLE 2.13: Tableau for Exercise 3

z	x_1	x_2	s_1	s_2	RHS
1	0	0	a	3	10
0	1	0	-1	-1	3
0	0	1	b	1	2

   (a) Suppose the current basic solution is not optimal but that the LP is bounded. After the next iteration of the simplex algorithm, what are the new values of the decision variables and objective function in terms of a and/or b?

   (b) Suppose the current basic solution is optimal but that the LP has alternative optimal solutions. Determine another such solution in terms of a and/or b.

   (c) If the LP is unbounded, what can be said about the signs of a and b?

4. Table 2.14 results after two iterations of the simplex algorithm are applied to a maximization problem having objective function, $z = f(x_1, x_2) = x_1 + 3x_2$.

TABLE 2.14: Tableau for LP having objective function $z = f(x_1, x_2) = x_1 + 3x_2$

z	x_1	x_2	s_1	s_2	RHS
1	0	0	-5	4	19
0	0	1	-1	1	5
0	1	0	-2	1	4

   Assume that the variables x_1 and x_2 are basic.

   (a) Explain why this LP is unbounded.

   (b) For each one unit increase in s_1, by how much will x_1 increase? By how much will x_2 increase?

   (c) Use your previous result to express x_2 as a linear function of x_1, valid for $s_1 \geq 0$.

(d) Sketch the current decision variable values as a point in the x_1x_2-plane. Draw a ray from this point, which falls upon the graph of the linear function from (c).

(e) What must be the values of x_1 and x_2 along this ray so that $z = 1000$? What is the corresponding value of s_1?

5. An LP involving maximization of a linear function in two decision variables has its feasible region shown in Figure 2.3.

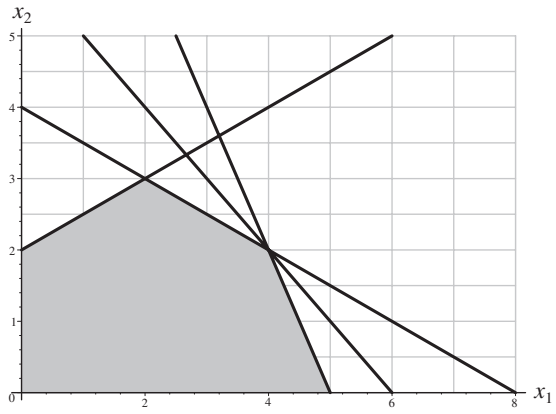


FIGURE 2.3: Feasible region for Exercise 5.

(a) List the basic solutions to this LP in vector form, $\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix}$, where $\mathbf{x}$ and $\mathbf{s}$ denote vectors of decision and slack variables, respectively. Explain why the LP is degenerate.

(b) What is the smallest possible number of iterations required to obtain a basic variable equal to zero? What is the largest possible number of iterations? What attribute of the objective function determines this outcome.

2.3 Excess and Artificial Variables: The Big M Method

In this section we describe a technique by which we can modify the simplex algorithm to solve LPs whose initial basic feasible solutions are not readily apparent. An example of such an LP is given by the *Foraging Herbivore Model*, from Exercise 4 of Section 1.1. For the sake of completeness, we reconstruct this model here.

In certain parts of the U.S., studies have shown that the vole, or common field mouse, is an herbivore whose diet consists predominantly of grass and a type of broad-leafed herb known as forb. Empirical studies suggest that the vole forages in a way that minimizes its total foraging time, subject to a set of two constraints. ([5])

Table 2.15 provides relevant information concerning digestive capacity and energy content. Food bulk records the extent to which the mass of a substance increases after it enters the digestive system and becomes liquid-saturated. For example, two grams of grass, when consumed, expands to $2 \times 1.64 = 3.28$ grams within the digestive system. To distinguish the mass of food prior to consumption from that in the digestive system, we use units of gm-dry and gm-wet, respectively.

TABLE 2.15: Data for *Foraging Herbivore* model

	Food bulk (gm-wet/gm-dry)	Energy content (kcal/gm)
Grass	1.64	2.11
Forb	2.67	2.30

The digestive capacity of the vole is 31.2 gm-wet per day, and the vole must consume enough food to meet an energy requirement of at least 13.9 kcal per day. Assume that the vole's foraging rate is 45.55 minutes per gram of grass and 21.87 minutes per gram of forb.

Let x_1 and x_2 denote the quantity of grass and forb, respectively, consumed by the vole on a given day. Units of both x_1 and x_2 are gm-dry, and consumption results in a total mass of $1.64x_1 + 2.67x_2$ gm-wet within the digestive system. Food bulk limitations then lead to the constraint

$$1.64x_1 + 2.67x_2 \leq 31.2.$$

Similar reasoning, based upon the daily energy requirement, yields $2.11x_1 + 2.3x_2 \geq 13.9$. Finally, the total foraging time, the quantity we seek to minimize, is given by $z = 45.55x_1 + 21.87x_2$. Combining these results, we arrive at the LP given in (2.6).

$$\begin{aligned} \text{minimize } z &= 45.55x_1 + 21.87x_2 \\ \text{subject to} & \\ 1.64x_1 + 2.67x_2 &\leq 31.2 \\ 2.11x_1 + 2.3x_2 &\geq 13.9 \\ x_1, x_2 &\geq 0. \end{aligned} \tag{2.6}$$

◆ ◆

Waypoint 2.3.1. Express LP (2.6) in the standard matrix form,

$$\begin{aligned} \text{maximize } z &= \mathbf{c} \cdot \mathbf{x} \\ \text{subject to} & \\ [A|I_3]\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\end{bmatrix} &= \mathbf{b} \\ \mathbf{x}, \mathbf{s} &\geq \mathbf{0}. \end{aligned} \tag{2.7}$$

Clearly specify the quantities, $\mathbf{c}$, A, and $\mathbf{b}$. The vector $\mathbf{b}$ should have one negative entry.

◆ ◆

In previously encountered LPs, an initial basic solution has been found by choosing decision variables as nonbasic and slack variables as basic. In the case of (2.7), this approach does not work. Specifically, if all decision variables are set to zero, entries of $\mathbf{s}$ are uniquely determined but at least one is negative due to the negative entry in $\mathbf{b}$. Hence, setting the decision variables equal to zero leads to a basic, but not basic feasible, solution. From a graphical perspective, as shown in Figure 2.4, the difficulty stems from the fact the origin lies outside the feasible region.

To investigate how the simplex algorithm can be modified to address this type of situation, we first consider the numerically simpler LP given by

$$\begin{aligned} \text{minimize } z &= 3x_1 + x_2 \\ \text{subject to} & \\ x_1 + x_2 &\leq 6 \\ x_1 &\geq 1 \\ 2x_1 + x_2 &= 10 \\ x_1, x_2 &\geq 0. \end{aligned} \tag{2.8}$$

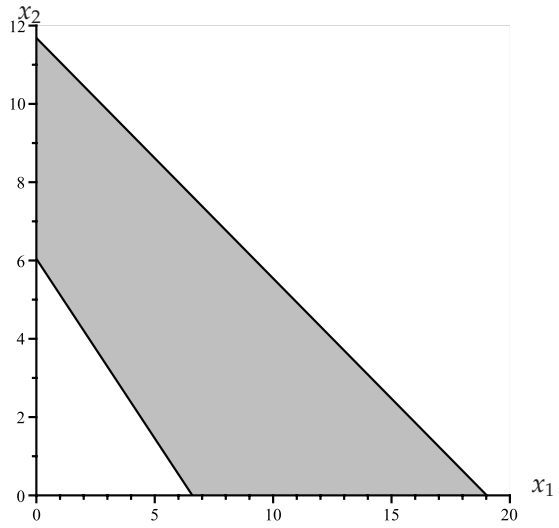


FIGURE 2.4: Feasible region for foraging herbivore LP, (2.6).

Note that LP (2.8) has an equality constraint; thus, its feasible region consists of a segment in $\mathbb{R}^2$.

Certainly we can introduce a slack variable, s_1 in the first constraint so that it becomes $x_1 + x_2 + s_1 = 6$. But rewriting the second constraint in $\leq$ form and adding a slack variable will only lead to the same problem of finding an initial basic feasible solution, so we instead subtract from x_1 a nonnegative *excess variable*, e_2, and rewrite the second constraint as $x_1 - e_2 = 1$. Clearly we can find nonnegative values of x_1 and e_2 that satisfy this equation, but determining nonnegative values of x_2 and s_1 that satisfy the other two constraint equations may still prove quite difficult.

To address this challenge, we introduce even more variables, called *artificial variables*, which are added to those constraints in which a $\geq$ or $=$ is present. To the second and third constraints, we introduce the nonnegative artificial variables a_2 and a_3, respectively, so that (2.8) becomes

$$\begin{aligned} &\text{minimize } z = 3x_1 + x_2 \\ &\text{subject to} \\ x_1 + x_2 + s_1 &= 6 \\ x_1 - e_2 + a_2 &= 1 \\ 2x_1 + x_2 + a_3 &= 10 \\ x_1, x_2, s_1, e_2, a_2, a_3 &\geq 0. \end{aligned} \tag{2.9}$$

Clearly this system yields a basic feasible solution in which the basic variables are given by $s_1 = 6$, $a_2 = 1$, and $a_3 = 10$.

The introduction of artificial variables therefore eliminates the difficulty of finding an initial basic feasible solution. However, these additional variables may yield an LP whose solution differs from that of the original LP. We therefore modify the objective function of the original LP so that all artificial variables in the new LP's final solution are ensured to equal zero. For if this outcome occurs and all artificial variables are zero, then the values of all remaining variables (decision, slack, and excess) are exactly the same as those in the optimal solution to the original LP.

The simplest way to modify the original objective function is to rewrite it as

$$z = 3x_1 + x_2 + Ma_2 + Ma_3,$$

where M is chosen to be a very large number. While there is no definitive rule as to how large M should be, it is important that M be significantly larger than any of the coefficients in the objective function, large enough so that any nonzero artificial variable value in the final solution results in a significantly larger z value. Put another way, the objective function incurs a substantial penalty of M units for every one unit increase in any artificial variable. Thus, we call this approach for solving the LP the *Big M Method*. For the particular example under discussion, we set $M = 20$. The initial tableau for the LP is given in Table 2.16.

TABLE 2.16: BV = $\{s_1, a_2, a_3\}$

z	x_1	x_2	s_1	e_2	a_2	a_3	RHS
1	-3	-1	0	0	-20	-20	0
0	1	1	1	0	0	0	6
0	1	0	0	-1	1	0	1
0	2	1	0	0	0	1	10

Since a_2 and a_3 are basic variables at the outset, pivots in their columns are necessary to record the initial objective function value. Performing two pivots yields the tableau in Table 2.17.

As this is a minimization LP, we focus on the column corresponding to the nonbasic variable whose coefficient in the top row of the tableau is most positive. This variable is x_1. By the ratio test, x_1 replaces a_2 as a basic variable, and we pivot on the indicated entry in Table 2.17. The resulting tableau is given in Table 2.18.

Two more iterations of the simplex algorithm are then necessary to achieve

TABLE 2.17: BV = $\{s_1, a_2, a_3\}$

z	x_1	x_2	s_1	e_2	a_2	a_3	RHS
1	57	19	0	-20	0	0	220
0	1	1	1	0	0	0	6
0	1	0	0	-1	1	0	1
0	2	1	0	0	0	1	10

TABLE 2.18: BV = $\{x_1, s_1, a_3\}$

z	x_1	x_2	s_1	e_2	a_2	a_3	RHS
1	0	19	0	37	-57	0	163
0	0	1	1	1	-1	0	5
0	1	0	0	-1	1	0	1
0	0	1	0	2	-2	1	8

the optimal solution. The intermediate and final tableaus are given in Tables 2.19 and 2.20.

TABLE 2.19: BV = $\{x_1, s_1, e_2\}$

z	x_1	x_2	s_1	e_2	a_2	a_3	RHS
1	0	$\frac{1}{2}$	0	0	-20	$-\frac{37}{2}$	15
0	0	$\frac{1}{2}$	1	0	0	$-\frac{1}{2}$	1
0	1	$\frac{1}{2}$	0	0	0	$\frac{1}{2}$	5
0	0	$\frac{1}{2}$	0	1	-1	$\frac{1}{2}$	4

Recall that because this is a standard minimization problem, we terminate the algorithm when entries in the top row of the tableau corresponding to nonbasic variables are all nonpositive. Thus the optimal solution is given by $(x_1, x_2) = (4, 2)$ and an objective value of $z = 14$. It is extremely important to observe that in this solution, both artificial variables are nonbasic!

To summarize, the Big M Method is a means of adapting the simplex algorithm to LPs involving a wider variety of constraints. We start with an LP written in a way that the right-hand side of each constraint is nonnegative. To each constraint involving a $\geq$, we subtract a positive excess variable and add an artificial variable. To each equality constraint, we simply add an artificial variable.

TABLE 2.20: BV = $\{x_1, x_2, e_2\}$

z	x_1	x_2	s_1	e_2	a_2	a_3	RHS
1	0	0	-1	0	-20	-18	14
0	0	1	2	0	0	-1	2
0	1	0	-1	0	0	1	4
0	0	0	-1	1	-1	1	3

Waypoint 2.3.2. Now use the Big M Method to solve the Herbivore Foraging Model (2.6).

Exercises Section 2.3

1. Solve each of the LPs using the Big M Method.

   (a)

$$\begin{aligned} \text{minimize } z &= -x_1 + 4x_2 \\ \text{subject to} & \\ x_1 + 3x_2 &\geq 8 \\ x_1 + x_2 &\geq 4 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

   (b)

$$\begin{aligned} \text{minimize } z &= x_1 + x_2 \\ \text{subject to} & \\ 2x_1 + 3x_2 &\geq 30 \\ -x_1 + 2x_2 &\leq 6 \\ x_1 + 3x_2 &\geq 18 \\ x_1, x_2 &\geq 0 \end{aligned}$$

(c)

$$\begin{aligned} &\text{maximize } z = x_1 + 5x_2 + 6x_3 \\ &\text{subject to} \\ &x_1 + 4x_2 + 2x_3 = 50 \\ &x_1 - 4x_2 + 4x_3 \geq 40 \\ &x_1, x_2, x_3 \geq 0 \end{aligned}$$

2.4 A Partitioned Matrix View of the Simplex Method

Partitioned matrices provide an elegant interpretation of the simplex algorithm in terms of matrix multiplication. This interpretation is crucial for the development of ideas in subsequent sections.

2.4.1 Partitioned Matrices

Simply put, a *partitioned matrix* is a "subdivision" of a matrix into smaller submatrices or "blocks," each obtained by deleting rows and/or columns of the original matrix. As a simple example, we consider the matrix M defined by

$$M = \begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 0 & -1 & 1 \\ -5 & 0 & 1 & 0 \\ 2 & 6 & 0 & 1 \end{bmatrix}. \tag{2.10}$$

Although M has four rows and columns, it can be partitioned into four, 2-by-2 matrices. Namely, if, $A = \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix}$, $C = \begin{bmatrix} -5 & 0 \\ 2 & 6 \end{bmatrix}$, and $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, then

$$M = \begin{bmatrix} A & B \\ C & I_2 \end{bmatrix}. \tag{2.11}$$

Clearly M can be partitioned in different ways. However, the goal in most contexts is to partition a matrix in a manner that facilitates performing matrix addition and/or multiplication.

For example, the product M^2 can be calculated directly, but it can also be computed by using (2.11) along with the fact all products involving A, B, C, and I_2 are defined. The result is as follows:

$$\begin{aligned} M^2 &= \begin{bmatrix} A & B \\ C & I_2 \end{bmatrix} \cdot \begin{bmatrix} A & B \\ C & I_2 \end{bmatrix} \\ &= \begin{bmatrix} A^2 + BC & AB + BI_2 \\ CA + I_2C & CB + I_2^2 \end{bmatrix} \\ &= \begin{bmatrix} A^2 + BC & AB + B \\ CA + C & CB + I_2 \end{bmatrix} \end{aligned} \tag{2.12}$$

Observe that we compute M^2 essentially by viewing each block as a scalar

and using the technique for multiplying 2-by-2 matrices. A simple calculation shows that (2.12), $A^2 + BC = \begin{bmatrix} -1 & 9 \\ 9 & 12 \end{bmatrix}$, which determines the four entries of M^2 belonging to the first two rows and first two columns. The other three entries of (2.12) are computed in a similar manner.

Waypoint 2.4.1. Calculate each of the quantities $AB + B$, $CA + C$, and $CB + I_2$. Use the results to compute the remaining entries of M^2. Then calculate M^2 directly using M itself to verify that your answer is correct.

Care must be exercised when multiplying partitioned matrices so as to ensure that all products involving submatrices in the partition are defined and computed in the correct order.

For example, suppose that A and B are 3-by-3 and 3-by-2 matrices, respectively. Assume that $\mathbf{c}$ is a (column) vector in $\mathbb{R}^3$, $\mathbf{d}$ is a (row) vector in $\mathbb{R}^2$, and that $\mathbf{0}_{3\times1}$ and $\mathbf{0}_{1\times3}$ represent the zero column vector and zero row vector in $\mathbb{R}^3$, respectively. Then $\begin{bmatrix} A & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}$ and $\begin{bmatrix} B & \mathbf{c} \\ \mathbf{d} & 2 \end{bmatrix}$ are 4-by-4 and 4-by-3 matrices, respectively, whose product is given by

$$\begin{bmatrix} A & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \cdot \begin{bmatrix} B & \mathbf{c} \\ \mathbf{d} & 2 \end{bmatrix} = \begin{bmatrix} AB + \mathbf{0}_{3\times1} \cdot \mathbf{d} & A\mathbf{c} + 2 \cdot \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} \cdot B + 1 \cdot \mathbf{d} & \mathbf{0}_{1\times3} \cdot \mathbf{c} + 2 \end{bmatrix} = \begin{bmatrix} AB & A\mathbf{c} \\ \mathbf{d} & 2 \end{bmatrix}. \tag{2.13}$$

One means of ensuring various matrix products are defined is to record dimensions of all matrices and vectors as subscripts. For example, in (2.13), the entry, $AB + \mathbf{0}_{3\times1} \cdot \mathbf{d}$, can be viewed as $A_{3\times3}B_{3\times2} + \mathbf{0}_{3\times1}\mathbf{d}_{1\times2}$, all of whose products and sums are defined and yield a 3-by-2 matrix.

2.4.2 Partitioned Matrices with Maple

Construction of partitioned matrices in Maple is very straightforward. Assume that matrices `A`, `B`, and `C` are defined using syntax discussed in Section C.5. Matrices are augmented through use of the | symbol and stacked using the comma. When augmentation and stacking are combined, the < and > signs are used for grouping purposes. To produce the partitioned matrix (2.11), we enter the following:

```
> A:=Matrix(2,2,[1,3,2,0]):
> B:=Matrix(2,2,[2,1,-1,1]):
> C:=Matrix(2,2,[-5,0,2,6]):
> I2:=IdentityMatrix(2):
> M:=<<A|B>,<C,I2>>;
```

2.4.3 The Simplex Algorithm as Partitioned Matrix Multiplication

Partitioned matrices offer an important perspective from which to view the simplex algorithm. We develop the main result by considering the *FuelPro* LP,

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} \leq \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{c} = \begin{bmatrix} 4 & 3 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 0 \\ 2 & 2 \\ 3 & 2 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}$. The initial tableau, as discussed in Section 2.1, is shown in Table 2.21.

TABLE 2.21: Initial tableau for *FuelPro* LP

z	x_1	x_2	s_1	s_2	s_3	RHS
1	-4	-3	0	0	0	0
0	1	0	1	0	0	8
0	2	2	0	1	0	28
0	3	2	0	0	1	32

As a partitioned matrix, Table 2.21 is represented by

$$\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{3\times 1} & A & I_3 & \mathbf{b} \end{bmatrix}. \tag{2.14}$$

We refer to (2.14) as the *tableau matrix* corresponding to Table (2.21), and we label its rows and columns so that the top row corresponds to row 0 and the leftmost column to column 0.

At each iteration of the simplex algorithm, we pivot on an entry in the tableau matrix. As indicated by Table 2.21, the initial pivot is on the entry in row 1, column 1. The first row operation of this pivot involves adding 4 times row 1 to row 0. If we let $\mathbf{y}$ denote a 1-by-3 row vector in $\mathbb{R}^3$ whose ith entry corresponds to the number of multiples of row i added to row 0, then $\mathbf{y} = [4, 0, 0]$ records

this first row operation. Moreover, after one iteration of the simplex algorithm, the new row 0, in matrix form, is given by

$$\begin{bmatrix} 1 - \mathbf{y} \cdot \mathbf{0}_{3\times 1} & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -3 & 4 & 0 & 0 & 32 \end{bmatrix}. \tag{2.15}$$

Waypoint 2.4.2. Verify that Equation (2.15) is indeed correct.

The remaining row operations needed to perform the pivot involve subtracting twice row 1 from row 2 and then three times row 1 from 3. Each of these row operations can be performed separately on the identity matrix to obtain $E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$, respectively.

Matrices, E_1 and E_2, are examples of *elementary matrices*, meaning that each is obtained by performing a single elementary row operation on the identity matrix. A fundamental property of any elementary matrix, E, is that left multiplication of a matrix B by E performs on B the elementary row operation corresponding to E. In the case of E_1 and E_2, left multiplication of B by the product $M = E_2E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$ has the overall effect of first subtracting twice row 1 from row 2 and then three times row 1 from 3. In a nutshell, when the submatrix of (2.14) obtained by deleting its top row is multiplied by M, the result obtained is precisely that needed to complete the pivot of the first simplex iteration. In the form of partitioned matrix multiplication,

$$\begin{aligned} M \cdot \begin{bmatrix} \mathbf{0}_{3\times 1} & A & I_3 & \mathbf{b} \end{bmatrix} &= \begin{bmatrix} M \cdot \mathbf{0}_{3\times 1} & MA & MI_3 & M\mathbf{b} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_{3\times 1} & MA & M & M\mathbf{b} \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 0 & 2 & -2 & 1 & 0 & 12 \\ 0 & 0 & 2 & -3 & 0 & 1 & 8 \end{bmatrix}. \end{aligned} \tag{2.16}$$

The results of (2.15) and (2.16) can now be in the following manner: We construct a partitioned matrix whose first column consists of $\begin{bmatrix} 1 \\ \mathbf{0}_{3\times 1} \end{bmatrix}$, i.e., column 0 of the tableau matrix, which remains constant from one iteration to the next. We then augment this first column on the right by the matrix $\begin{bmatrix} \mathbf{y} \\ M \end{bmatrix}$, which re-

sults in $\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0}_{3\times 1} & M \end{bmatrix}$. This matrix is then right-multiplied by the original tableau matrix as follows:

$$\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0}_{3\times 1} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{3\times 1} & A & I_3 & \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{c}+\mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0}_{3\times 1} & MA & M & M\mathbf{b} \end{bmatrix}. \tag{2.17}$$

The result of (2.17) is the tableau matrix obtained after one simplex algorithm iteration is applied to the *FuelPro* LP. A Maple worksheet **Simplex Algorithm as Partitioned Matrix Multiplication.mw**, useful for verifying this fact is given as follows.

```
> restart;with(LinearAlgebra):
> c:=Vector[row]([4,3]);
  # Create row vector with objective coefficients.
```

$$c := [4, 3]$$

```
> A:=Matrix(3,2,[1,0,2,2,3,2]);
  # Matrix of constraint coefficients.
```

$$A := \begin{bmatrix} 1 & 0 \\ 2 & 2 \\ 3 & 2 \end{bmatrix}$$

```
> b:=<8,28,32>;
  # Constraint bounds.
```

$$b := \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}$$

```
> TableauMatrix:=<<UnitVector(1,4) | <-c,A>  | <ZeroVector[row](3),
  IdentityMatrix(3)>  | <0,b>>;
  # Create initial tableau matrix.
```

$$TableauMatrix := \begin{bmatrix} 1 & -4 & -3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 2 & 2 & 0 & 1 & 0 & 28 \\ 0 & 3 & 2 & 0 & 0 & 1 & 32 \end{bmatrix}$$

```
> y:=[4,0,0];
```

$$y := Vector[row]([4, 0, 0])$$

```
> M:=Matrix(3,3,[1,0,0,-2,1,0,-3,0,1]);
```

$$M := \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$

```
> S:=<<<1>|y>,<ZeroVector(3)|M>>;
  # Create partitioned matrix using y and M, which corresponds
  to first simplex iteration.
```

$$S := \begin{bmatrix} 1 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -3 & 0 & 1 \end{bmatrix}$$

```
> S.TableauMatrix;
  # Result after first iteration of simplex algorithm.
```

$$\begin{bmatrix} 1 & 0 & -3 & 4 & 0 & 0 & 32 \\ 0 & 1 & 0 & 1 & 0 & 0 & 8 \\ 0 & 0 & 2 & -2 & 1 & 0 & 12 \\ 0 & 0 & 2 & -3 & 0 & 1 & 8 \end{bmatrix}$$

We now consider the second simplex iteration. Relabel $\mathbf{y}$ and M in (2.17) as $\mathbf{y}_1$ and M_1. The second iteration introduces a new 3-by-1 vector, $\mathbf{y}_2$, and a new 3-by-3 matrix, M_2, that record row operations needed to perform the second iteration. The tableau matrix obtained after the second iteration is then given as follows:

$$\begin{aligned} &\begin{bmatrix} 1 & \mathbf{y}_2 \\ \mathbf{0}_{3\times 1} & M_2 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & \mathbf{y}_1 \\ \mathbf{0}_{3\times 1} & M_1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{3\times 1} & A & I_3 & \mathbf{b} \end{bmatrix} \right) \\ &\quad = \left(\begin{bmatrix} 1 & \mathbf{y}_2 \\ \mathbf{0}_{3\times 1} & M_2 \end{bmatrix} \cdot \begin{bmatrix} 1 & \mathbf{y}_1 \\ \mathbf{0}_{3\times 1} & M_1 \end{bmatrix} \right) \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{3\times 1} & A & I_3 & \mathbf{b} \end{bmatrix} \\ &\quad = \begin{bmatrix} 1 & \mathbf{y}_1 + \mathbf{y}_2 M_1 \\ \mathbf{0}_{3\times 1} & M_2 M_1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{3\times 1} & A & I_3 & \mathbf{b} \end{bmatrix} \\ &= \begin{bmatrix} 1 & -\mathbf{c} + (\mathbf{y}_1 + \mathbf{y}_2 M_1)A & \mathbf{y}_1 + \mathbf{y}_2 M_1 & (\mathbf{y}_1 + \mathbf{y}_2 M_1)\mathbf{b} \\ \mathbf{0}_{3\times 1} & M_2 M_1 A & M_2 M_1 & M_2 M_1 \mathbf{b} \end{bmatrix}. \end{aligned} \tag{2.18}$$

◆ ◆

Waypoint 2.4.3. For the *FuelPro* LP, use the elementary row operations performed at the second iteration of the simplex algorithm to verify that

$$\mathbf{y}_2 = \begin{bmatrix} 0 & 0 & \frac{3}{2} \end{bmatrix} \quad \text{and } M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}. \tag{2.19}$$

Then combine these results with the previously computed values of $\mathbf{y}_1$ and M_1 to verify that (2.18) coincides with the tableau matrix corresponding to Table 2.3 from Section 2.1.

◆ ◆

Clearly this process of expressing each tableau as a product of partitioned matrices continues to further iterations. We summarize the results in Theorem 2.4.1.

Theorem 2.4.1. Consider the LP written in matrix inequality form as

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} \leq \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where A is an m by n matrix, $\mathbf{c}$ and $\mathbf{x}$ are row and column vectors, respectively, in $\mathbb{R}^n$, and $\mathbf{b}$ belongs to $\mathbb{R}^m$. Row operations corresponding to the kth iteration of the simplex algorithm are characterized by left multiplication using a partitioned (m+1) by (m+1) matrix of the form

$$\begin{bmatrix} 1 & \mathbf{y}_k \\ \mathbf{0}_{m\times 1} & M_k \end{bmatrix}, \tag{2.20}$$

where $\mathbf{y}_k$ is a row vector in $\mathbb{R}^m$ and M_k is an m by m matrix. The tableau matrix after the kth iteration is the corresponding matrix after the $(k-1)^{\text{st}}$ iteration, left multiplied by (2.20). This inductive process leads to the following representation of the tableau matrix resulting from the kth iteration:

$$\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0}_{m\times 1} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times m} & 0 \\ \mathbf{0}_{m\times 1} & A & I_3 & \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{c}+\mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0}_{m\times 1} & MA & M & M\mathbf{b} \end{bmatrix}, \tag{2.21}$$

where

$$M = M_k \cdot M_{k-1} \cdots M_1 \text{ and } \mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2 M_1 + \mathbf{y}_3 M_2 M_1 + \ldots + \mathbf{y}_k M_k M_{k-1} \cdots M_1.$$

For the *FuelPro* LP, $\mathbf{y}_1 = \begin{bmatrix} 4 & 0 & 0 \end{bmatrix}$, $\mathbf{y}_2 = \begin{bmatrix} 0 & 0 & \frac{3}{2} \end{bmatrix}$, $M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$, and $M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$. It then follows that for $k = 2$,

$$\begin{aligned} \mathbf{y} &= \mathbf{y}_1 + \mathbf{y}_2 M_1 \\ &= \begin{bmatrix} 4 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -\frac{1}{2} & 0 & \frac{3}{2} \end{bmatrix} \end{aligned}$$

and

$$M = M_2M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & -1 \\ -\frac{3}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

Using these results, we can form the partitioned matrix $\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0}_{m\times 1} & M \end{bmatrix}$, which, in turn, provides a means for determining the tableau matrix after the second iteration.

At this stage, Theorem 2.4.1 is merely a theoretical result, one that does not affect the manner in which we perform the simplex algorithm. However, it demonstrates that the quantities $\mathbf{y}$ and M alone in (2.21) can be used with an LP's initial tableau matrix to determine the entire tableau matrix at a given iteration. In the next section we exploit this fact and improve the efficiency of the simplex algorithm.

Exercises Section 2.4

1. Let $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Compute the product AB by partitioning both A and B in two different ways.

2. Suppose that A is a 2-by-3 matrix, C and E are row vectors in $\mathbb{R}^3$ and $\mathbb{R}^2$, respectively, B and F are 2-by-2 matrices. Let $M = \begin{bmatrix} A & B \\ C & E \end{bmatrix}$ and $N := \begin{bmatrix} A^t \\ F \end{bmatrix}$, where A^t denotes the transpose of A. Calculate MN in terms of A, B, C, E, and F.

3. Suppose that a and b are scalars, $\mathbf{x}$ and $\mathbf{y}$ are row vectors in $\mathbb{R}^3$, $\mathbf{u}$ and $\mathbf{v}$ are (column) vectors in $\mathbb{R}^3$, and A and B are 3-by-3 matrices. Let $M = \begin{bmatrix} a & \mathbf{x} \\ \mathbf{v} & A \end{bmatrix}$ and $N = \begin{bmatrix} b & \mathbf{y} \\ \mathbf{u} & B \end{bmatrix}$. Calculate MN in terms of a, b, $\mathbf{x}$, $\mathbf{y}$, $\mathbf{u}$, $\mathbf{v}$, A, and B.

4. Suppose that A and B are n by n matrices and that 0_n denotes the n by n zero matrix. Show that A and B are invertible if and only if $M = \begin{bmatrix} A & 0_n \\ 0_n & B \end{bmatrix}$ is invertible.

5. For each of parts (a) and (c) of Exercise 1 in Section 2.1, calculate the vector, $\mathbf{y}_1$, and matrix, M_1, corresponding to the first iteration of the simplex algorithm.

6. For the *FuelPro* LP, calculate the vector, $\mathbf{y}_3$, and matrix, M_3, corresponding to the third and final iteration of the simplex algorithm. Then, calculate the final tableau matrix, using the vectors and matrices, M_1, M_2, M_3, $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$.

2.5 The Revised Simplex Algorithm

When executed by hand, the simplex algorithm requires writing numerous tableaus, many of whose values remain unchanged from one iteration to the next. The purpose of this section is to address this shortcoming of the algorithm.

2.5.1 Notation

Throughout this section we shall again refer to the standard maximization problem

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} \leq \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where A is an m-by-n matrix, $\mathbf{c}$ and $\mathbf{x}$ belong to $\mathbb{R}^n$, and $\mathbf{b}$ belongs to $\mathbb{R}^m$. We will assume that all entries of $\mathbf{b}$ are positive. The revised simplex algorithm builds upon the result of Theorem 2.4.1 from Section 2.4. Namely, the tableau matrix after the kth iteration, in partitioned matrix multiplication form, is given by

$$\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0} & MA & M & M\mathbf{b} \end{bmatrix}, \tag{2.22}$$

where $\mathbf{y}$ is a row vector in $\mathbb{R}^m$ and M is an m-by-m matrix. The vector $\mathbf{y}$ and matrix M record all row operations used to obtain the kth tableau matrix. To underscore the fact that these quantities correspond to the kth iteration, we modify our notation and relabel $\mathbf{y}$ and M from Theorem 2.4.1 and (2.22) as $\mathbf{y}_k$ and M_k, respectively. We also define the $m+1$-by-$m+1$ matrix

$$S_k = \begin{bmatrix} 1 & \mathbf{y}_k \\ \mathbf{0} & M_k \end{bmatrix} \qquad k = 0, 1, 2, \ldots \tag{2.23}$$

With this notation, the tableau matrix after the kth iteration is the product of S_k and the initial tableau matrix:

$$S_k \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix}. \tag{2.24}$$

Note that when $k = 0$, $\mathbf{y}_0 = \mathbf{0}_{1\times m}$, in which case $S_0 = I_{m+1}$, the $m+1$-by-$m+1$ identity matrix. Hence, when $k = 0$, (2.24) is merely the original tableau matrix.

2.5.2 Observations about the Simplex Algorithm

The route to the revised algorithm begins with the following observations.

1. After the kth iteration of the simplex algorithm, we identify the entering nonbasic variable by focusing on the nonbasic variable coefficients in the top row of the tableau. One means of identifying the nonbasic variable whose coefficient is most negative is to multiply the matrix S_k by each column vector of the initial tableau matrix that corresponds to a nonbasic variable. However, only the first entry in each matrix-vector product is needed to identity the entering nonbasic variable.

2. The entering nonbasic variable and the basic variable it replaces determine a pivot entry in the column corresponding to the entering variable. The other entries in this column vector alone dictate the elementary operations required to update the tableau. Furthermore, in columns of the tableau corresponding to variables that remain basic, all entries below row 0 remain unchanged after the iteration is complete.

3. The tableau matrix after the $(k+1)^{\text{st}}$ iteration is obtained by applying to (2.24) the previously described row operations. The matrix S_{k+1} is therefore the result of applying these row operations to S_k.

In the subsequent discussions, we will use the following notation. We let BV_k, where $0 \leq k \leq m$, be an $m+1$ by 1 column vector of "labels," whose top entry is z and whose jth entry is the basic variable associated with row j of the tableau after the kth iteration. For example, in the *FuelPro* LP, $\text{BV}_1 = \begin{bmatrix} z \\ x_1 \\ s_2 \\ s_3 \end{bmatrix}$ since the basic variables after the first iteration are given by x_1 (row 1), s_2 (row 2), and s_3 (row 3).

2.5.3 An Outline of the Method

Using this notation, our algorithm can be outlined in the following steps.

1. At the start, BV_0 is made up of the initial basic variables in the order corresponding to their rows (the slack variables in our discussion), and $S_0 = I_{m+1}$.

2. At the kth stage, we first compute $S_k \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$ and then multiply S_k times the column vector corresponding to each nonbasic variable in the initial tableau matrix. After computing only the top entry in each matrix-vector product, we can identify the entering variable, if one exists, as the one corresponding to the most negative top entry. (If all such top entries are

nonnegative, then the algorithm terminates and the final tableau matrix is given by (2.24).) We then compute the remaining entries in the vector corresponding to only the entering variable. Call this vector **v**.

3. We apply the ratio test, just as in the usual simplex algorithm, using entries in $S_k \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$ and the vector **v** computed in the previous step. (We ignore, however, the ratio in the top entry, which corresponds to row 0 of the tableau matrix.) The result identifies the pivot entry in **v**. Knowing the pivot entry in **v**, we have identified the variable that will become nonbasic. This allows us to compute BV_{k+1}.

4. Because **v** corresponds to the column of the entering basic variable, we must pivot on the entry identified in the previous step. The entries of **v** alone dictate the row operations required to carry out this process. These row operations are performed on S_k to create S_{k+1}.

5. We now return to step (2).

2.5.4 Application to the *FuelPro* LP

To illustrate an application of this algorithm, we return to the *FuelPro* LP.

TABLE 2.22: Initial tableau for *FuelPro* Petroleum problem

z	x_1	x_2	s_1	s_2	s_3	RHS
1	-4	-3	0	0	0	0
0	1	0	1	0	0	8
0	2	2	0	1	0	28
0	3	2	0	0	1	32

Recall that our convention is to number the top row of the tableau matrix, excluding labels, as row 0 and the leftmost column as column 0.

1. Initially, $\text{BV}_0 = \begin{bmatrix} z \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$ and $S_0 = I_4$.

2. We have that

$$\text{BV}_0 = S_0 \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \\ 28 \\ 32 \end{bmatrix}. \tag{2.25}$$

Since the nonbasic variables are x_1 and x_2, we compute the top entries in

the matrix-vector products of S_0 and the corresponding initial tableau matrix column vectors:

$$S_0 \begin{bmatrix} -4 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -4 \\ \blacksquare \\ \blacksquare \\ \blacksquare \end{bmatrix} \quad \text{and} \quad S_0 \begin{bmatrix} -3 \\ 0 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ \blacksquare \\ \blacksquare \\ \blacksquare \end{bmatrix},$$

so that x_1 is the entering variable and

$$\mathbf{v} = S_0 \begin{bmatrix} -4 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -4 \\ 1 \\ 2 \\ 3 \end{bmatrix}. \tag{2.26}$$

3. Using the results of (2.25) and (2.26), we compute the respective ratios as 8, 14, and $\frac{32}{3}$. The result tells us that x_1 replaces s_1 in the second entry (i.e., row 1 entry) of BV_0 so that $\text{BV}_1 = \begin{bmatrix} z \\ x_1 \\ s_2 \\ s_3 \end{bmatrix}$.

4. We now pivot on the second entry of $\mathbf{v} = \begin{bmatrix} -4 \\ 1 \\ 2 \\ 3 \end{bmatrix}$. This process requires the following three elementary row operations:

   (a) Add 4 times row 1 to row 0;

   (b) Add -2 times row 1 to row 2;

   (c) Add -3 times row 1 to row 3.

   To compute S_1, we perform on $S_0 = I_4$ these elementary row operations, in the order they are listed. The result is as follows:

$$S_1 = \begin{bmatrix} 1 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & -3 & 0 & 1 \end{bmatrix}.$$

   This completes the first iteration of the revised method.

5. We begin the second iteration by computing $S_1 \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}$. One means of performing this task is of course to carry out the matrix-vector multiplication. Another way is to perform on $S_0\mathbf{b}$, the same elementary row

operations that derived S_1 from S_0. Either approach leads to

$$S_1 \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} 32 \\ 8 \\ 12 \\ 8 \end{bmatrix}. \tag{2.27}$$

The nonbasic variables are x_2 and s_1. Thus we compute only the top entries in the matrix-vector products of S_0 and the corresponding initial tableau matrix column vectors:

$$S_1 \begin{bmatrix} -3 \\ 0 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ \blacksquare \\ \blacksquare \\ \blacksquare \end{bmatrix} \quad \text{and} \quad S_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ \blacksquare \\ \blacksquare \\ \blacksquare \end{bmatrix},$$

so that x_2 is the entering variable and

$$\mathbf{v} = S_1 \begin{bmatrix} -3 \\ 0 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 2 \\ 2 \end{bmatrix}. \tag{2.28}$$

6. The results of (2.27), (2.28), and the ratio test tell us to pivot on the fourth entry (i.e., on the row 3 entry) in the result from (2.28). Thus x_2 replaces s_3 as a basic variable and

$$\mathrm{BV}_2 = \begin{bmatrix} z \\ x_1 \\ s_2 \\ x_2 \end{bmatrix}.$$

7. The required row operations to pivot on the row 3 entry of (2.28) are as follows:

   (a) Add the negative of row 3 to row 2.

   (b) Add $\frac{3}{2}$ of row 3 to row 0.

   (c) Scale row 3 by $\frac{1}{2}$.

   These row operations, when applied to S_1, yield

$$S_2 = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & \frac{3}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 \\ 0 & -\frac{3}{2} & 0 & \frac{1}{2} \end{bmatrix},$$

   thereby completing the second iteration.

An important observation to make regarding this new notation is that the matrix S_k is a submatrix of the tableau matrix obtained after the kth iteration. It is formed using columns of the tableau matrix corresponding to the slack variables, augmented on the left with the column vector, $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$. Indeed, the matrices S_1 and S_2 can easily be identified as submatrices of Tables 2.2 and 2.3, respectively, from Section 2.1.

◆ ◆

Waypoint 2.5.1. Execute a third iteration of the revised simplex algorithm for the *FuelPro* LP. Your results should indicate that $\text{BV}_3 = \begin{bmatrix}z\\x_1\\s_1\\x_2\end{bmatrix}$ and $S_3\mathbf{b} = \begin{bmatrix}46\\4\\4\\10\end{bmatrix}$. Begin to carry out a fourth iteration. You should discover that when S_3 is multiplied by either column vector of the tableau matrix corresponding to a nonbasic variable, the top entry of the vector is positive. Hence the algorithm terminates with $\text{BV}_3 = S_3\mathbf{b}$ so that $z = 46$, $x_1 = 4$, $s_1 = 4$ and $x_2 = 10$.

◆ ◆

In practice, the revised simplex algorithm has a significant advantage over the original algorithm in that it requires less memory storage, specifically that needed to record the large number of zeros and ones in columns corresponding to basic variables.

Exercises Section 2.5

Solve each of the following LPs using the revised simplex algorithm.

1.

$$\begin{aligned} &\text{maximize } z = 3x_1 + 2x_2 \\ &\text{subject to} \\ &x_1 \leq 4 \\ &x_1 + 3x_2 \leq 15 \\ &2x_1 + x_2 \leq 10 \\ &x_1, x_2 \geq 0 \end{aligned}$$

2.

$$\begin{aligned} &\text{maximize } z = 4x_1 + x_2 + 5x_3 \\ &\text{subject to} \\ &2x_1 + x_2 + 3x_3 \leq 14 \\ &6x_1 + 3x_2 + 3x_3 \leq 22 \\ &2x_1 + 3x_2 \leq 14 \\ &x_1, x_2, x_3 \geq 0 \end{aligned}$$

3.

$$\begin{aligned} &\text{minimize } z = 3x_1 - 2x_2 \\ &\text{subject to} \\ &x_1 - 2x_2 \leq 2 \\ &x_1 + x_2 \leq 4 \\ &x_1, x_2 \geq 0 \end{aligned}$$

2.6 Moving beyond the Simplex Method: An Interior Point Algorithm

The revised simplex algorithm represents a slight improvement on the original method. However, it does not address a major shortcoming of the simplex algorithm, namely that the number of iterations can grow exponentially large with the size of the problem. In this section, we investigate a radically different approach for solving LPs.

2.6.1 The Origin of the Interior Point Algorithm

The *FuelPro* LP is a small-scale problem having only two decision variables, three constraints, and two sign restrictions. If we start with the initial basic feasible solution at the origin, we arrive at the optimal solution after only three iterations of the simplex algorithm. Obviously, for larger scale problems, the number of iterations needed to obtain an optimal solution can become quite large. How much so was demonstrated in 1972, when Klee and Minty constructed a family of LPs, each having n decision variables and n constraints, where $n = 1, 2, 3, \ldots$, yet requiring 2^n simplex algorithm iterations to solve [19]. In simple terms, the number of iterations needed to solve an LP can be exponential in the problem size. However, this is an extreme case, and for most practical applications, far fewer iterations are required.

In 1984, Narendra Karmarkar, an employee of AT&T, devised a method for obtaining an approximate solution to any LP, one that achieved any desired degree accuracy in a number of iterations that was no greater than a polynomial in the problem size [18], [10]. His method differs fundamentally from the simplex algorithm in that it begins with a feasible solution of the LP having the property that every variable value, be it decision, slack, or excess, is strictly positive. Such a point is said to belong to the *interior* of the feasible region. Through an iterative process, a sequence of feasible solutions is constructed, which is contained in the feasible region's interior and which converges to the optimal solution.

2.6.2 The Projected Gradient

Karmarkar's original method is beyond the scope of this text, so we instead focus on a simple variant known as the *affine scaling algorithm*. Throughout the discussion, we consider a maximization problem, but to avoid the need for notation that distinguishes among variable types (e.g., decision, slack, excess), we express the LP in a form having only equality-type constraints.

Namely, we write the LP as

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} = \mathbf{b} \\ &\qquad \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.29}$$

where $\mathbf{x}$ belongs to $\mathbb{R}^n$, $\mathbf{c}$ is a row vector in $\mathbb{R}^n$, $\mathbf{b}$ belongs to $\mathbb{R}^m$, and A is an m-by-n matrix. For the *FuelPro* LP, $\mathbf{x}$ belongs to $\mathbb{R}^5$,

$$\mathbf{c} = \begin{bmatrix} 4 & 3 & 0 & 0 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}, \text{ and } A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix}.$$

In this notation, x_3, x_4, and x_5 correspond to the slack variables from the original *FuelPro* LP, and the matrix A denotes the coefficient matrix from the original *FuelPro* LP, augmented with the identity matrix, I_3.

To begin the algorithm, we start with a point, $\mathbf{x}_0$ that belongs to the interior of the LP's feasible region. To say that $\mathbf{x}_0$ is feasible means that $A\mathbf{x}_0 = \mathbf{b}$. That it belongs to the interior means each component of $\mathbf{x}_0$ is strictly positive, as opposed to merely nonnegative. For example, $\mathbf{x}_0 = \begin{bmatrix} 3 \\ 4 \\ 5 \\ 14 \\ 15 \end{bmatrix}$ satisfies both these conditions for the *FuelPro* LP. Finally, we say that $\mathbf{x}_0$ belongs to the boundary of the feasible region if at least one of its components is zero.

We now start at $\mathbf{x}_0$ and seek a new point $\mathbf{x}_1 = \mathbf{x}_0 + \triangle\mathbf{x}$, whose objective, $\mathbf{c}\mathbf{x}_1$, is larger than at $\mathbf{x}_0$. An initial choice for $\triangle\mathbf{x}$ is $\mathbf{c}^t$, the transpose vector of $\mathbf{c}$. That this choice makes sense can be seen if we observe that $\nabla(\mathbf{c} \cdot \mathbf{x}) = \mathbf{c}^t$, the gradient of z, which "points" in the direction of greatest increase in z. (We discuss the gradient in much greater detail in Part II of the text.) However, this choice does not work if we use the feasibility of $\mathbf{x}_0$, along with the desired feasibility of $\mathbf{x}_1$. Namely,

$$\begin{aligned} \mathbf{b} &= A\mathbf{x}_1 \\ &= A\left(\mathbf{x}_0 + \mathbf{c}^t\right) \\ &= A\mathbf{x}_0 + A\mathbf{c}^t \\ &= \mathbf{b} + A\mathbf{c}^t. \end{aligned}$$

This result implies that $\Delta\mathbf{x} = \mathbf{c}^t$ must be a solution of the homogeneous matrix equation, $A\mathbf{x} = \mathbf{0}$, or, in other words, $\Delta\mathbf{x} = \mathbf{c}^t$ belongs to the null space of the

matrix A from (2.29). Such a requirement does not hold for the case of the *FuelPro* LP, nor for any LP in which the column vectors of A form a linearly independent set.

To circumvent this problem, we must therefore choose $\Delta\mathbf{x}$ so that it "points" as much as possible in the direction of $\mathbf{c}^t$, yet belongs to the null space of A. One means of accomplishing this task is to project $\mathbf{c}^t$ onto the null space of A by means of a matrix transformation.

Definition 2.6.1. Suppose an LP in m constraints and n variables is written in the form (2.29), where A is an m-by-n matrix. Then the *projected gradient* associated with the LP is defined as the quantity

$$\mathbf{c}_p = P\mathbf{c}^t, \tag{2.30}$$

where $P = \left(I_n - A^t\left(AA^t\right)^{-1}A\right)$.

In this definition, we assume that the m-by-m matrix AA^t is invertible. This is true for the *FuelPro* LP and any matrix A having linearly independent row vectors.

◆ ——— ◆

Waypoint 2.6.1. For the *FuelPro* LP, calculate the projection matrix P. Then use your result to verify that the projected gradient is given by

$$\mathbf{c}_p = \begin{bmatrix} \frac{6}{35} \\ \frac{1}{7} \\ -\frac{6}{35} \\ -\frac{22}{25} \\ -\frac{4}{5} \end{bmatrix}.$$

◆ ——— ◆

The projected gradient, $\mathbf{c}_p$, and corresponding projection matrix P satisfy three important properties, whose verifications we leave as exercises:

1. The vector $\mathbf{c}_p$ belongs to the null space of A. That is, $A\mathbf{c}_p = \mathbf{0}$.
2. The matrix P is symmetric, meaning $P^t = P$.
3. The matrix P satisfies $P^2\mathbf{c}^t = P\mathbf{c}^t$.

The following theorem is a consequence of these facts.

Theorem 2.6.1. For $\alpha > 0$, the objective evaluated at $\mathbf{x}_1 = \mathbf{x}_0 + \alpha\mathbf{c}_p$ is at least as large as that evaluated $\mathbf{x}_0$. In other words, $\mathbf{c}\mathbf{x}_1 \geq \mathbf{c}\mathbf{x}_0$.

Proof. Using the preceding properties, along with properties of the transpose, we have

$$\begin{aligned}
\mathbf{c}\cdot\mathbf{x}_1 - \mathbf{c}\cdot\mathbf{x}_0 &= \mathbf{c}\left(\mathbf{x}_1 - \mathbf{x}_0\right)\\
&= \mathbf{c}\cdot\alpha\mathbf{c}_p\\
&= \alpha\mathbf{c}\left(P\mathbf{c}^t\right)\\
&= \alpha\mathbf{c}\left(P^2\mathbf{c}^t\right)\\
&= \alpha\left(\mathbf{c}P^t\right)\left(P\mathbf{c}^t\right)\\
&= \alpha\left(P\mathbf{c}^t\right)^t\left(P\mathbf{c}^t\right)\\
&= \alpha\left\|\mathbf{c}_p\right\|^2\\
&\geq 0.
\end{aligned}$$

□

2.6.3 Affine Scaling

Theorem 2.6.1 demonstrates that for $\alpha > 0$, $\mathbf{x}_1 = \mathbf{x}_0 + \alpha\mathbf{c}_p$ has an objective value at least as large as that of $\mathbf{x}_0$ and that this value increases as α increases. Since $\mathbf{c}_p$ belongs to the null space of A, $A\mathbf{x}_1 = A\mathbf{x}_0 + \alpha A\mathbf{c}_p = A\mathbf{x}_0\mathbf{b}$. However, this fact alone does not guarantee that $\mathbf{x}_1$ is feasible. For if α increases by too much, there exists the possibility that a component of $\mathbf{x}_1$ becomes negative, in which case $\mathbf{x}_1$ does not satisfy the sign restrictions. The key then is to allow α to increase, but not by too much.

The *FuelPro* LP illustrates the delicacy of this situation. Using $\mathbf{x}_0 = \begin{bmatrix} 3\\4\\5\\14\\15 \end{bmatrix}$ and the result from the previous Waypoint, we have

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 + \alpha\mathbf{c}_p \qquad (2.31)\\
&= \begin{bmatrix} 3\\4\\5\\14\\15 \end{bmatrix} + \alpha\begin{bmatrix} \frac{6}{35}\\ \frac{1}{7}\\ -\frac{6}{35}\\ -\frac{22}{25}\\ -\frac{4}{5} \end{bmatrix}\\
&= \begin{bmatrix} 3+\frac{6}{35}\alpha\\ 4+\frac{1}{7}\alpha\\ 5-\frac{6}{35}\alpha\\ 14-\frac{22}{25}\alpha\\ 15-\frac{4}{5}\alpha \end{bmatrix}.
\end{aligned}$$

The last component of this vector dictates that α can increase by no more than $18\frac{3}{4}$ before $\mathbf{x}_1$ violates the sign restrictions.

Setting $\alpha = 18\dfrac{3}{4}$ results in

$$\mathbf{x}_1 = \begin{bmatrix} \frac{87}{14} \\ \frac{187}{28} \\ \frac{25}{14} \\ \frac{31}{14} \\ 0 \end{bmatrix}, \tag{2.32}$$

with corresponding objective value $z = \mathbf{c} \cdot \mathbf{x}_1 = \frac{1257}{28} \approx 44.9$.

This objective value is much larger than $z = \mathbf{c} \cdot \mathbf{x}_0 = 24$ and much closer to the known optimal value of $z = 46$. Moreover, the last component of $\mathbf{x}_1$ is zero, implying that $\mathbf{x}_1$ corresponds to a point on the boundary of the original LP's feasible region where the third constraint, $3x_1 + 2x_2 \leq 32$, is binding. However, the values of the original LP's decision variables, $x_1 = \dfrac{87}{14} \approx 6.21$, and $x_2 = \dfrac{187}{28} \approx 6.67$, are far from close to the known optimal solution, $x_1 = 4$ and $x_2 = 10$.

Repeating this entire process by computing a new value $\mathbf{x}_2$ from $\mathbf{x}_1$ in the same manner we computed $\mathbf{x}_1$ from $\mathbf{x}_0$ will not improve the quality of the estimate. This follows from observing that the last component of $\mathbf{x}_2 = \mathbf{x}_1 + \alpha \mathbf{c}_p$ is $-\dfrac{4}{5}\alpha$. For $\mathbf{x}_2$ to remain feasible, α must equal zero, whence $\mathbf{x}_2 = \mathbf{x}_1$.

This result demonstrates a fundamental problem we must overcome, which can be summarized in the following terms. The quantity α can be viewed as a "step size," which measures what fraction of the projected gradient, $\mathbf{c}_p$, we use to obtain the next approximation, $\mathbf{x}_{k+1}$, from the current one, $\mathbf{x}_k$. On the one hand, we desire a large step size so that the difference in objective values, $\mathbf{c}\mathbf{x}_{k+1} - \mathbf{c}\mathbf{x}_k = \alpha \mathbf{c} \cdot \mathbf{c}_p$, is significant. On the other hand, if the step size, α, is too large, a component of $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{c}_p$ becomes negative so that $\mathbf{x}_{k+1}$ violates the sign restrictions.

The key to overcoming this problem is to perform a change of variables at each iteration. To compute $\mathbf{x}_{k+1}$, we first use the previous iterates, $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$, to create a completely new LP. In this new LP, the initial value, $\tilde{\mathbf{x}}_0$, which corresponds to $\mathbf{x}_k$ in the original LP, lies far enough from the boundary of the feasible region so that if we set it to the initial value in new LP, we are permitted to select a large step size. After using this large step size to calculate a better approximate solution to the new LP, we "change back" variables, so to speak, to determine the corresponding value of $\mathbf{x}_{k+1}$.

The change of variables we use is a simple matrix transformation involving an invertible, diagonal matrix, D, which we define momentarily. The commuting diagram shown in Figure 2.5 illustrates our strategy.

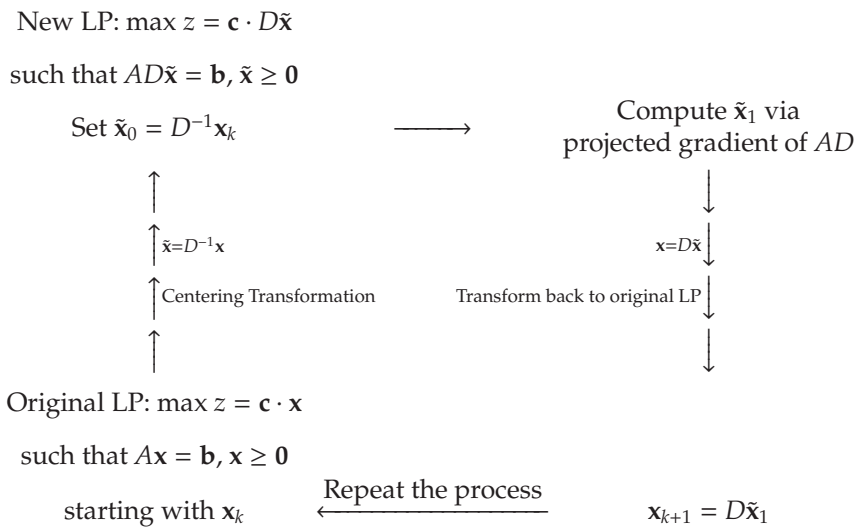


FIGURE 2.5: Commuting diagram illustrating change of variables.

The matrix D that we elect to use has the effect of "centering" $\tilde{\mathbf{x}_k}$ in the feasible region of the new LP. It is an n-by-n matrix defined by

$$D = \begin{bmatrix} x_{k,1} & 0 & \dots & 0 \\ 0 & x_{k,2} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & x_{k,n} \end{bmatrix}, \tag{2.33}$$

where $x_{k,i}, 1 \leq i \leq n$ denotes the ith component of $\mathbf{x}_k$.

Here are two important facts regarding the matrix D.

- Since every component of $\mathbf{x}_k$ is strictly positive, D is invertible and

$$D^{-1} = \begin{bmatrix} x_{k,1}^{-1} & 0 & \dots & 0 \\ 0 & x_{k,2}^{-1} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & x_{k,n}^{-1} \end{bmatrix}$$

  .

- In the new LP, each component of $\tilde{\mathbf{x}_0} = D^{-1}\mathbf{x}_k$ equals one. This means that each component of $\tilde{\mathbf{x}_0}$ is one unit from the boundary of the interior of the new LP. To emphasize this crucial fact, we use the notation, $\mathbf{1}_n$ to denote the vector $\tilde{\mathbf{x}_0}$.

2.6.4 Summary of the Method

Here then is an overview of our interior point algorithm.

1. Suppose $\mathbf{x}_k$, where $k \geq 0$, lies in the interior of the feasible region of the original LP, meaning $A\mathbf{x}_k = \mathbf{b}$ and each component of $\mathbf{x}_k$ is strictly positive.

2. We construct a diagonal matrix, D, whose diagonal values are the components of $\mathbf{x}_k$.

3. In the new LP, the objective coefficient is given by $\tilde{\mathbf{c}} = \mathbf{c}D$ and the coefficient matrix by $\tilde{A} = AD$. We compute the projected gradient of $\tilde{\mathbf{c}}$ for the new LP. That is, we project $\tilde{\mathbf{c}}$ onto the null space of $\tilde{A}$ and set

$$\tilde{\mathbf{c}_p} = P\tilde{\mathbf{c}}^t, \tag{2.34}$$

   where

$$P = \left(I_n - \tilde{A}^t\left(\tilde{A}\tilde{A}^t\right)^{-1}\tilde{A}\right).$$

4. Since $\tilde{\mathbf{x}}_0 = \mathbf{e}$, where $\mathbf{e}$ is the vector in $\mathbb{R}^n$ all of whose entries are one, α can be at most the reciprocal of the absolute value of the most negative entry of $\tilde{\mathbf{c}}_p$ before an entry of $\tilde{\mathbf{x}}_0 + \alpha\tilde{\mathbf{c}}_p$ becomes negative. We therefore set α to equal this quantity.

5. Define
$$\tilde{\mathbf{x}}_1 = \mathbf{e} + \lambda\alpha\tilde{\mathbf{c}}_p,$$
where λ is a fixed quantity slightly less than one. The value of $\lambda\alpha$ controls the stepsize used in the new LP.

6. Let $\mathbf{x}_{k+1} = D\tilde{\mathbf{x}}_1$, the $(k+1)$st iterate in the original LP.

7. Return to Step 1 and repeat the process.

2.6.5 Application of the Method to the *FuelPro* LP

We now demonstrate the details for performing one iteration of the algorithm for the *FuelPro* LP.

1. We start with the initial value $\mathbf{x}_0 = \begin{bmatrix} 3 \\ 4 \\ 5 \\ 14 \\ 15 \end{bmatrix}$.

2. Using $\mathbf{x}_0$,
$$D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 15 \end{bmatrix}$$
.

3. In the new LP, $\tilde{\mathbf{c}} = \mathbf{c}D = \begin{bmatrix} 12 & 12 & 0 & 0 & 0 \end{bmatrix}$ and $\tilde{A} = AD = \begin{bmatrix} 3 & 0 & 5 & 0 & 0 \\ 6 & 8 & 0 & 14 & 0 \\ 9 & 8 & 0 & 0 & 15 \end{bmatrix}$. These quantities are then used to construct the projection matrix, P, and projected gradient, $\tilde{\mathbf{c}}_p$. The second of these is given by $\tilde{\mathbf{c}}_p \approx \begin{bmatrix} 4.57 \\ 5.85 \\ -2.74 \\ -5.30 \\ -5.86 \end{bmatrix}$.

4. The last entry of $\tilde{\mathbf{c}}_p$ indicates that $\alpha = 1/5.86 \approx .1706$, the reciprocal of

the absolute value of most negative component of $\tilde{\mathbf{c}}_p$. Using this value along with $\lambda = .75$, we compute $\tilde{\mathbf{x}}_1$:

$$\begin{aligned}\tilde{\mathbf{x}}_1 &= \tilde{\mathbf{x}}_0 + \lambda\alpha\tilde{\mathbf{c}}_p \\ &\approx \begin{bmatrix}1.58\\1.75\\.65\\.32\\.25\end{bmatrix}\end{aligned}$$

5. In the original LP,

$$\mathbf{x}_1 = D\tilde{\mathbf{x}}_1 \approx \begin{bmatrix}4.75\\6.99\\3.25\\4.50\\3.75\end{bmatrix},$$

which has corresponding objective value, $\mathbf{c} \cdot \mathbf{x}_1 \approx 40$.

2.6.6 A Maple Implementation of the Interior Point Algorithm

The affine scaling method described in this section can be implemented quite easily using Maple. What follows is a worksheet, **Interior Point Algorithm.mw**, that computes the four iterations.

```
> restart;with(LinearAlgebra):
> c:=Vector[row]([4,3,0,0,0]);
  # Create row vector with objective coefficients.
```

$$c := [4, 3, 0, 0, 0]$$

```
> A:=Matrix(3,5,[1,0,1,0,0, 2,2,0,1,0, 3,2,0,0,1]);
  # Matrix of constraint coefficients.
```

$$A := \begin{bmatrix}1 & 0 & 1 & 0 & 0\\2 & 2 & 0 & 1 & 0\\3 & 2 & 0 & 0 & 1\end{bmatrix}$$

```
> b:=<8,28,32>;
  # Constraint bounds.
```

$$b := \begin{bmatrix}8\\28\\32\end{bmatrix}$$

```
> N:=4:lambda:=.75:
  # Set number of iterations, N, and parameter lambda.
```

```
> x:=array(0..N):
  # Array of iteration values.

> x[0]:=<3,4,5,14,15>;
  # Set initial value.
```

$$x_0 := \begin{bmatrix} 3 \\ 4 \\ 15 \\ 14 \\ 15 \end{bmatrix}$$

```
> for i from 0 to (N-1) do
  d:=DiagonalMatrix(convert(x[i],list)):
  # Determine transformation matrix.
  ~ A:= A.d: ~ c:=c.d:
  # Calculate coefficient matrix and objective coefficients for
  new LP.
  P:=IdentityMatrix(5)-
  Transpose(~ A).MatrixInverse(~ A.Transpose(~ A)).~ A:
  # The projection matrix.
  cp:=P.Transpose(~ c):
  # Determine projected gradient.
  alpha:=(abs(min(seq(cp[j],j=1..5))))^(-1):
  # Find alpha.
  x[i+1]:=d.(<1,1,1,1,1>+lambda*alpha*cp):
  # Determine next iterate.
  end do:

> for i from 0 to N do print(x[i],Transpose(c).x[i]):end do:
  # Print sequence of iterates, together with corresponding objective
  values.
```

For the sake of brevity, we do not print here the output that appears at the end of this worksheet. Instead, we summarize these results in Table 2.23 but list only the entries of each $\mathbf{x}_i$ that correspond to the two decision variables, x_1 and x_2, in the original *FuelPro* LP.

TABLE 2.23: Results of applying the interior point algorithm to the *FuelPro* LP

n	$(x_{n,1}, x_{n,2})$	z
0	$(3, 4)$	24
1	$(4.75, 7)$	40
2	$(5.1, 7.95)$	44.07
3	$(4.66, 8.9)$	45.32
4	$(4.1, 9.79)$	45.76

The results in Table 2.23 dictate that convergence to the optimal solution takes place very quickly. A graphical illustration of this phenomenon is depicted in Figure 2.6.

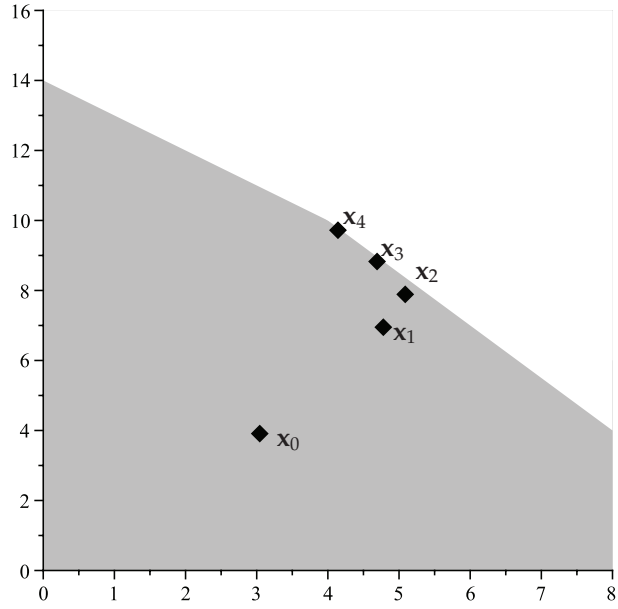


FIGURE 2.6: Interior algorithm iterates for *FuelPro* LP.

Since 1984, interior point methods have continually evolved and improved in their efficiency. Interestingly, after a great deal of controversy centered on the issuance of patents for what could be viewed as newly developed mathematical tools, AT& T applied for, and received a patent for Karmarkar's method in 1988. This patent expired in 2006.

Exercises Section 2.6

1. Suppose that A is an m-by-n matrix whose rows form a linearly independent set of vectors. Show that the m-by-m matrix AA^t is invertible. (Hint: Start with the homogeneous matrix equation $AA^t\mathbf{x} = \mathbf{0}$. Multiply each side by $\mathbf{x}^t$, the transpose vector of $\mathbf{x}$. Use the result to establish $\left\|A^t\mathbf{x}\right\| = 0$, and then apply the Invertible Matrix Theorem.)

2. Show that the projection matrix, P, and projected gradient, $\mathbf{c}_p$, as given in Definition 2.6.1 satisfy the following three properties:
   (a) The vector $\mathbf{c}_p$ belongs to the null space of A. That is, $A\mathbf{c}_p = \mathbf{0}$.
   (b) The matrix P is symmetric, meaning that $P^t = P$.
   (c) The matrix P satisfies $P^2\mathbf{c}^t = P\mathbf{c}^t$.

3. Assume in the *FuelPro* LP that $\mathbf{x}_0 = \begin{bmatrix} 3 \\ 4 \\ 5 \\ 14 \\ 15 \end{bmatrix}$, which corresponds to $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$ in x_1x_2-plane. If we denote $\tilde{\mathbf{x}}_k = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$, sketch the image in the $\tilde{x}_1\tilde{x}_2$-plane of each basic feasible solutions of the *FuelPro* LP under the first centering transformation D^{-1}. Show that the region in the $\tilde{x}_1\tilde{x}_2$-plane enclosed by these points coincides with the feasible region of the LP,

$$\begin{aligned} &\text{maximize } z = \tilde{\mathbf{c}}\tilde{\mathbf{x}} \\ &\text{subject to} \\ &\tilde{A}\tilde{\mathbf{x}} \leq \mathbf{b} \\ &\tilde{\mathbf{x}} \geq \mathbf{0}, \end{aligned}$$

   where $\tilde{A}$ and $\tilde{\mathbf{c}}$ are as defined in step 3 of Section 2.6.5.

4. Approximate, to three decimal places, the solutions of the following LPs.

   (a)

$$\begin{aligned} &\text{maximize } z = 3x_1 + 2x_2 \\ &\text{subject to} \\ &x_1 \leq 4 \\ &x_1 + 3x_2 \leq 15 \\ &2x_1 + x_2 \leq 10 \\ &x_1, x_2 \geq 0 \end{aligned}$$

   (b)

$$\begin{aligned} &\text{maximize } z = 4x_1 + x_2 + 5x_3 \\ &\text{subject to} \\ &2x_1 + x_2 + 3x_3 \leq 14 \\ &6x_1 + 3x_2 + 3x_3 \leq 22 \\ &2x_1 + 3x_2 \leq 14 \\ &x_1, x_2, x_3 \geq 0 \end{aligned}$$

5. Use the interior point algorithm to approximate the solution of the *Foraging Herbivore Model*, from Exercise 4 of Section 1.1.

Chapter 3

Standard Applications of Linear Programming

3.1 The Diet Problem

Within the field of linear programming, there exist certain classes of closely related problems. In the vast majority of them, efficiently formulating the LP itself so that it can be solved using Maple or other available software is a challenge. In this chapter, we investigate several problem types.

3.1.1 Eating for Cheap on a Very Limited Menu

A sandwich shop has a menu consisting of four items: turkey breast on a bun, a club sandwich, a veggie sub, and a breakfast sandwich. Nutritional data for each is summarized in Table 3.1.

TABLE 3.1: Sandwich nutritional data

Sandwich	Turkey Breast	Club	Veggie Sub	Breakfast Sandwich
Calories	280	320	230	470
Fat (gm.)	4.5	6	3	19
Cholesterol (gm.)	20	35	0	200
Carbohydrates (gm.)	46	47	44	53
Fiber (gm.)	5	5	5	5
Protein (gm.)	18	24	9	28
Vitamin A (pct. RDA)	8	8	8	10
Vitamin C (pct. RDA)	35	35	35	15
Calcium (pct. RDA)	6	8	6	25
Iron (pct. RDA)	25	30	25	25

Daily nutritional guidelines for each category in Table 3.1 are summarized in Table 3.2. The listed minimum requirements for vitamins A and C, calcium, and iron are based upon the assumption that deficiencies are remedied through beverages and daily vitamin supplements.

Suppose that sandwiches cost $6.00 (turkey breast), $5.00 (club sandwich),

TABLE 3.2: Daily nutritional guidelines

Category	Daily Minimum	Daily Maximum
Calories	2000	3000
Fat (gm.)	0	65
Cholesterol (gm.)	0	300
Carbohydrates (gm.)	50	300
Fiber (gm.)	25	50
Protein (gm.)	50	200
Vitamin A (pct. RDA)	40	200
Vitamin C (pct. RDA)	40	200
Calcium (pct. RDA)	40	200
Iron (pct. RDA)	40	200

$3.50 (veggie sub), and $5.00 (breakfast sandwich). We seek to minimize the total daily cost of maintaining a diet that is comprised solely of these four sandwiches and fulfills the previously stated nutritional guidelines. We will permit partial consumption of any sandwich and assume that the purchase price of any such sandwich is pro-rated on the basis of the fraction consumed. For example, consumption of one-half of a club will contribute only $2.50 to total cost. Certainly this last assumption is unrealistic in that the number of purchased sandwiches of each type must be integer-valued. In Chapter 5, we will learn how to account for this fact.

3.1.2 The Problem Formulation and Solution, with Help from Maple

The LP whose solution minimizes total cost in this situation will have numerous constraints, and listing these constraints individually within Maple's `LPSolve` can prove extremely tedious. We therefore seek a more efficient manner for constructing them. Key to doing so are Maple's array, sequence, and adding commands as introduced in Section C.4. What follows is merely one of many means of combining these commands in a manner that leads to the correct LP solution.

We first define decision variables, x_1, x_2, x_3, and x_4, to be the number of consumed turkey breast sandwiches, club sandwiches, veggie subs, and breakfast sandwiches, respectively. Since these decision variables will appear in our objective function and throughout our constraints, we define them in Maple through use of an array as follows:

```
> restart: with(LinearAlgebra):with(Optimization):
> x:=array(1..4);
```

$$x := array(1..4, [])$$

Each column of Table 3.1 can be viewed as a vector in $\mathbb{R}^{10}$ recording nutritional information for a particular sandwich. One means of storing this information in Maple is to first construct a "nutrition vector" for each sandwich and then to use the resulting vectors to form a "nutrition matrix." The following syntax performs this task:

```
> TurkeyBreast:=<280, 4.5, 20, 46, 5, 18, 8, 35, 6, 25>:
> Club:=<320,6,35,47,5,24,8,35,8,30>:
> VeggieSub:=<230,3,0,44,5,9,8,35,6,25>:
> BreakfastSandwich:=<470,19,200,53,5,28,10,15,25,25>:
> A:=<TurkeyBreast | Club | VeggieSub | BreakfastSandwich>:
```

In a similar manner, each column of Table 3.2 can also be entered as a vector in $\mathbb{R}^{10}$. The terms `Maximum`, `maximum`, `max`, as well as corresponding terms for minima, have reserved meanings in Maple, so we must choose names for these vectors with care. Here we call them `MinimumAmount` and `MaximumAmount`. The first of these is given as follows:

```
> MinimumAmount :=<2000,0,0,50,25,50,40,40,40,40>:
```

The vector `MaximumAmount` is defined similarly.

While the total cost, in dollars, can be expressed in Maple as `6*x[1]+5*x[2]+3.5*x[3]+5*x[4]`, we will enter sandwich costs in the form of a vector. Doing so illustrates a convenient tool for larger-scale problems.

```
> Prices:=<6,5,3.5,5>:
> TotalCost:=add(Prices[i]*x[i],i=1..4);
```

$$TotalCost := 6x_1 + 5x_2 + 3.5x_3 + 5x_4$$

Constraints can be formulated in terms of the array, `x`, the matrix, `A`, and the vectors `MinimumAmount` and `MaximumAmount`. For example, the maximum number of calories can be expressed as follows:

```
> add(A[1,j]*x[j],j=1..4)<=3000;
```

$$280x_1 + 320x_2 + 230x_3 + 470x_4 \leq 3000$$

We wish to ensure that analogous guidelines hold for other categories. One means of accomplishing this efficiently in Maple is through use of the sequence command, `seq`. What follows is syntax illustrating how this command can be used to produce all model constraints:

```
> MinimumGuidelines:=
  seq(MinimumAmount[i]<=add(A[i,j]*x[j],j=1..4),i=1..10):

> MaximumGuidelines:=
  seq(add(A[i,j]*x[j],j=1..4)<=MaximumAmount[i],i=1..10):
```

These preceding two command lines produce the LP constraints in the form of two sequences, whose results can be used to create the constraint list for purposes of invoking the `LPSolve` command. Here is the command line that combines the previous results to determine the optimal solution of the LP.

```
> LPSolve(TotalCost,[MinimumGuidelines,MaximumGuidelines],
  assume='nonnegative');
```

Waypoint 3.1.1. Use Maple to verify that the cheapest, four-sandwich diet costs approximately $29.81 per day and consists of no turkey breast sandwiches, approximately 5.14 club sandwiches, .314 veggie subs, and .6 breakfast sandwiches.

Maple structures of sequences, sums, vectors, matrices, and arrays all played a role in solving the diet LP in this section. It is important to note that in many problems more than one structure can be used to perform a task. For example, `Prices` was entered as a vector but, for purposes of computing total cost, could also be entered as a list, `Prices:=[6,5,3.5,5]`. Choosing this approach would not affect subsequent command lines. A matrix was chosen over an array to record nutritional data for all sandwiches merely due to the fact nutritional information for each sandwich was presented in tabular form (as one would expect from reading product labels) and could be easily entered as a vector. These vectors were combined to produce the matrix `A`. In other problem types, use of an array is more appropriate. When confronted with choices such as these, two important points to be bear in mind are the following: Which method permits the easiest entry of data relevant to the problem, and which method permits the greatest flexibility in terms of needed operations for constructing the LP? For example, a `Matrix` can be multiplied by a column `Vector`, but not by a list structure, unless the list is first converted to a `Vector` through use of the `convert` command.

Exercises Section 3.1

1. Most nutritional guidelines stipulate that sodium intake be limited daily to 2400 milligrams. Sodium amounts for the four sandwiches are given in Table 3.3. Show that if the limit on daily sodium intake is incorporated into the four-sandwich diet model, then the LP becomes infeasible. Estimate the minimum daily sodium intake for which a feasible solution exists.

TABLE 3.3: Sandwich sodium amounts

Sandwich	Sodium Amount (mg.)
Turkey Breast	1000
Club	1290
Veggie Sub	500
Breakfast Sandwich	1500

2. Suppose we desire to follow a modified four-sandwich diet that seeks to minimize carbohydrate intake subject to fulfilling nutritional guidelines for only protein, vitamin A, vitamin C, calcium, and iron while consuming at least 40 grams of fiber per day. What four-sandwich diet accomplishes this goal? What is the corresponding carbohydrate intake?

3.2 Transportation and Transshipment Problems

Many linear programming problems focus on cost-effective ways to transport goods and services of one type or another.

3.2.1 A Coal Distribution Problem

Three coal mines transport coal to four different municipal power plants. Suppose the mines, which we label as Mine 1, 2, and 3, have respective annual productions of 500,000; 500,000; and 400,000 tons. The four municipalities, Cities 1, 2, 3, and 4, have corresponding annual demands of 400,000; 300,000; 500,000; and 200,000 tons. To each combination of a mine and city, there is a cost per unit amount associated with transporting coal from the mine to the city. Table 3.4 summarizes costs, in units of millions of dollars per hundred thousand tons, for each different mine-city combination.

TABLE 3.4: Mine-city transportation costs

Mine/City	City 1	City 2	City 3	City 4
Mine 1	2	3	2	4
Mine 2	4	2	2	1
Mine 3	3	4	3	1

We seek to construct an LP whose objective is to minimize the total cost of transporting the coal from the mines to the cities. Because the total coal amount produced by the mines equals the combined demand of the cities, this LP constitutes a *balanced transportation problem*. Clearly if the total demand exceeds total available supply, any LP model attempting to minimize cost will prove infeasible. The case when total supply exceeds total demand is addressed in the exercises.

We begin by defining x_{ij}, where $1 \le i \le 3$ and $1 \le j \le 4$, as the amount of coal, in hundreds of thousands of tons, transported from Mine i to City j. Because of the double subscript nature of these decision variables, we enter them as an array in Maple:

```
> x:=array(1..3,1..4);
```

$$array(1..3, 1..4, [])$$

If the entries of Table 3.4 are also entered as a 3-by-4 array (matrix), labeled `Cost`, the total transportation cost, in millions of dollars, is computed as follows:

```
> Cost:=Matrix(3,4,[2,3,2,4,4,2,2,1,3,4,3,1]):
> TotalCost:=add(add(Cost[i,j]*x[i,j],i=1..3),j=1..4);
```

$$TotalCost := 2x_{1,1} + 3x_{1,2} + 2x_{1,3} + 4x_{1,4} + 4x_{2,1} + 2x_{2,2} + 2x_{2,3} + x_{2,4} + 3x_{3,1} + 4x_{3,2} + 3x_{3,3} + x_{3,4}$$

As was the case for the Four-Sandwich Diet Problem from Section 3.1, the `seq` and `add` commands, can be combined to ensure that supply and demand constraints are met. For example, suppose that the cities' respective demand amounts are defined by the list (or array or vector) named `Demand`. Then `seq(add(x[i,j],i=1..3) >=Demand[j], j=1..4)` is a sequence of four inequalities, which, if satisfied, ensures that demand is met.

Waypoint 3.2.1. In a manner similar to that used for demand, define a list, `Supply`, and use it to construct a sequence of inequalities that reflects the amount of supply available at each mine. Combine this sequence with that for demand to form a list of constraints for this transportation LP. Then solve the LP using Maple's `LPSolve` command. Immediately after the `LPSolve` command, enter the following commands, which assigns the decision variables their solution values and prints these values in tabular form. Your results should agree with what appears below.

```
> assign(%[2]): # Assign the decision variables to
  their solution values.
> print(x): # Print the solution array in tabular
  format.
```

$$\begin{bmatrix} 2 & 0 & 3 & 0 \\ 0 & 3 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

3.2.2 The Integrality of the Transportation Problem Solution

The solution of the Coal Distribution Problem has the property that all decision variables are integer-valued. This is a trait we desire of LPs in a wide variety of contexts. In many situations we are forced to use the techniques of integer linear programming, which form the basis of discussion in Chapter 5. Fortunately, the nature of the Coal Distribution Problem, in particular the form of its constraints, guarantees, a priori, an integer-valued solution.

To explain this phenomenon in terms of matrix theory, we start with the general transportation problem consisting of m supply points and n demand points. Let $\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}$ and $\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}$ denote vectors in $\mathbb{R}^m$ and R^n, whose entries record the supply amounts and demand amounts, respectively, at each of the supply points and demand points. Assume further that entries in these vectors are all integer-valued and that the problem is *balanced*, meaning $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$. If x_{ij}, where $1 \le i \le m$ and $1 \le j \le n$, denotes the amount of goods delivered from supply point i to demand point j and if c_{ij} denotes the corresponding cost, then the balanced transportation problem may be stated as

$$\begin{aligned} &\text{minimize } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (3.1)\\ &\text{subject to}\\ &\quad \sum_{j=1}^{n} x_{ij} = s_i \quad 1 \le i \le m\\ &\quad \sum_{i=1}^{m} x_{ij} = d_j \quad 1 \le j \le n\\ &\quad x_{ij} \ge 0 \text{ for } 1 \le i \le m \text{ and } 1 \le j \le n. \end{aligned}$$

Observe that we may express each constraint as an equality due to the balanced nature of the LP.

The constraints of LP (3.1) can be written compactly if we introduce new notation. More specifically, let us define $\mathbf{0}_{1\times n}$ and $\mathbf{1}_{1\times n}$ to be the row vectors in $\mathbb{R}^n$ of all zeros and all ones, respectively. Then the constraints of (3.1) can be expressed in partitioned matrix equation form as follows:

$$m \text{ blocks}\left\{\vphantom{\begin{bmatrix}a\\b\\c\\d\\e\end{bmatrix}}\right. \overbrace{\begin{bmatrix} \mathbf{1}_{1\times n} & \mathbf{0}_{1\times n} & \mathbf{0}_{1\times n} & \cdots & \mathbf{0}_{1\times n} \\ \mathbf{0}_{1\times n} & \mathbf{1}_{1\times n} & \mathbf{0}_{1\times n} & \cdots & \mathbf{0}_{1\times n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{1\times n} & \mathbf{0}_{1\times n} & \ddots & \mathbf{0}_{1\times n} & \mathbf{1}_{1\times n} \\ I_n & I_n & \cdots & I_n & I_n \end{bmatrix}}^{m \text{ blocks}} \cdot \begin{bmatrix} x_{11} \\ \vdots \\ x_{1n} \\ x_{21} \\ x_{22} \\ \vdots \\ x_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{s} \\ \mathbf{d} \end{bmatrix}. \qquad (3.2)$$

The $(m + n)$ by mn coefficient matrix in (3.2) takes a very special form that guarantees each of its square submatrices has determinant equal to 0 or ± 1. The proof of this fact uses basic determinant properties, and we do not include it here. ([14])

Because the LP is balanced, the equation corresponding to the bottom row of matrix equation (3.2) is merely the sum of the equations corresponding to the first m rows, less the sum of those corresponding to rows $m + 1$ through $m + n$. It follows that (3.2) can be rewritten as

$$A\mathbf{x} = \mathbf{b}, \tag{3.3}$$

where A is an $m + n - 1$ by mn matrix, $\mathbf{x}$ is a vector in $\mathbb{R}^{mn}$ consisting of mn decision variables, and $\mathbf{b}$ is a vector $\mathbb{R}^{m+n-1}$ having integer-valued entries.

Our standard procedure for calculating a basic solution of (3.1) requires us to set $mn - (m + n - 1) = (m - 1)(n - 1)$ decision variables to zero and solve for the remaining basic variables, assuming that the columns of A corresponding to the basic variables form a linearly independent set. Of course, the square matrix formed using these columns is an invertible submatrix of the coefficient matrix in (3.2), and hence must have determinant equal to $\pm = 1$. We leave it as an exercise to show that any matrix equation, $A\mathbf{x} = \mathbf{b}$, in which A and $\mathbf{b}$ have integer entries and $\det A = \pm 1$, leads to a solution vector $\mathbf{x}$ whose entries are also integer-valued. Thus, we arrive at the following theorem.

Theorem 3.2.1. The decision variable values in any basic solution of a transportation problem are integer-valued, provided the problem is balanced and the demand and supply amounts are themselves integer-valued. Hence, the decision variable values in the optimal solution of such an LP are also integer-valued.

3.2.3 Coal Distribution with Transshipment

We now consider a variant on our coal shipment problem by adding two intermediate points between the mines and the cities. In the context of this problem, these intermediate points, sometimes referred to as *transshipment* points, can be thought of as railyard facilities where shipments from different mines are combined and then delivered to each of the four cities. In this situation, there exists a cost per unit amount to transfer coal from each mine to each transshipment point and from each transshipment point to each city. These costs, in millions of dollars per one hundred thousand tons, are summarized in Tables 3.5 and 3.6, where we have labeled the transshipment points as Station 1 and Station 2.

To minimize the cost of transporting the coal from the mines to the cities, we view the problem as one combining two separate transportation problems. In the first of these, transshipment points are viewed as "demand points"; in the

TABLE 3.5: Mine-railyard transportation costs

Mine/Station	Station 1	Station 2
Mine 1	4	3
Mine 2	5	4
Mine 3	2	4

TABLE 3.6: Railyard-city transportation costs

Station/City	City 1	City 2	City 3	City 4
Station 1	3	5	4	3
Station 2	4	3	4	4

second, they constitute "supply points." Combined with these two problems are constraints, sometimes referred to as *conservation constraints*, which reflect the requirement that the amount of coal entering a transshipment point is the same as the amount leaving.

In a manner similar to that for the original problem, we use arrays to record the shipment amounts between the various points. We define x_{ij} where $1 \leq i \leq 3$ and $1 \leq j \leq 2$, as the amount of coal transported from Mine i to Station j. Similarly, y_{jk}, where $1 \leq j \leq 2$ and $1 \leq k \leq 4$, denotes the amount transported from Station j to City k. In Maple we have:

```
> x:=array(1..3,1..2):
> y:=array(1..2,1..4):
```

Entries from Tables 3.5 and 3.6 can be entered as two matrices, which we label `CostTo` and `CostFrom`. Using them, we construct the total transportation cost as follows:

```
> CostTo:=Matrix(3,2,[4,3,5,4,2,4]):
> CostFrom:=Matrix(2,4,[3,5,4,3,4,3,4,4]):
> TotalCost:=add(add(CostTo[i,j]*x[i,j],i=1..3),j=1..2)+
  add(add(CostFrom[j,k]*y[j,k],j=1..2),k=1..4):
```

Each constraint for the LP belongs to three one of the three categories:

1. The supply at each mine is the sum of the amounts transported from that mine to each of the stations.

2. The demand at each city is the sum of the amounts transported to that city from each of the stations.

3. Conservation constraints that guarantee the total amounts transported into and out of each station are equal.

Using the arrays `x` and `y` and the `Demand` and `Supply` lists defined in the first problem, we express these constraints in Maple as follows:

```
> SupplyConstraints:=seq(add(x[i,j],j=1..2) <=Supply[i],i=1..3):
> DemandConstraints:=seq(add(y[j,k],j=1..2) >=Demand[k],k=1..4):
> NetFlowConstraints:=
  seq(add(x[i,j],i=1..3)=add(y[j,k],k=1..4),j=1..2):
```

Waypoint 3.2.2. Use the previously defined sequences to form a list of constraints and determine the entries of `x` and `y` that minimize total costs in this transshipment model.

Exercises Section 3.2

1. For the coal model without transshipment, suppose that demand at City 3 decreases from 500,000 to 300,000 tons, in which case the problem is no longer balanced. Determine the least costly manner in which to deliver coal from the mines to cities and still meet demand. Where will the excess coal remain under this scenario? (Hint: Define an "imaginary" City 5, whose demand is the excess of total supply over total demand, i.e., 200,000 tons. Set to zero the transportation cost from each mine to City 4.)

2. Prove the assertion immediately prior to Theorem 3.2.1. Namely, if A is a square, n by n having determinant equal to ± 1 and if both A and $b \in \mathbb{R}^n$ have integer-valued entries, then the solution vector of $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$ also has integer-valued entries. (Hint: Use Cramer's Rule. See Appendix B.)

3. *Import substitution* refers to a nation's practice of producing a certain commodity domestically so as to reduce the amount of the commodity imported from abroad. [1] Suppose a country desires to reduce its dependence upon overseas sources for corn and soy. The country has three available plantations for producing a mixed crop of these items, which are then transported to four different markets. The plantations are labeled 1, 2, and 3, and have 9, 10, and 10 acres available, respectively, for growing either crop. The demand for corn and soy at each of the four markets is summarized in Table 3.7.

[1] Based upon Blandford, Boisvert, and Charles [8], (1982).

TABLE 3.7: Market demand for corn and soy, measured in tons

Market/Commodity	Corn	Soy
1	2	5
2	5	8
3	10	13
4	17	20

Each plantation produces 2 tons of corn per acre and 4 tons of soy per acre. The costs of growing corn and soy are \$200 per acre and \$300 per acre, respectively, regardless of the plantation. The cost of transporting either good from plantation i to market j is $20i + 30j$ dollars per ton.

(a) Suppose crop 1 represents corn and crop 2 represents soy. Let x_{ijk}, where $1 \leq i \leq 3$, $1 \leq j \leq 4$, and $1 \leq k \leq 2$, represent the amount of crop k, measured in tons, that is transported from plantation i to market j. Let y_{ik}, where $1 \leq i \leq 3$, and $1 \leq k \leq 2$, represent the acreage at plantation i that is devoted to producing crop k. Use this notation to determine an expression that represents the total cost due to growing the two crops on the three plantations and transporting them to the different markets.

(b) Determine a sequence of inequalities that stipulates the acreage supply at each plantation is not exceeded.

(c) Determine a second sequence of inequalities that guarantees demand at each market is met.

(d) Determine a third sequence of inequalities that guarantees the amount of each crop, in tons, transported from a given plantation to a given market does not exceed the available amount of the crop that can be produced at that plantation.

(e) Determine the values of the quantities x_{ijk} and y_{jk} that minimize total cost due to import substitution of corn and soy.

3.3 Basic Network Models

Transportation and transshipment problems are special cases of the *minimum cost network flow problem*, which we now describe in its general form.

3.3.1 The Minimum Cost Network Flow Problem Formulation

The minimum cost network flow problem arises from a collection of *n nodes* each pair of which is connected by a *directed arc*. To the arc starting at node i and ending at node j, where $1 \leq i, j \leq n$ we may associate two quantities, an arc *flow capacity*, M_{ij}, and a *cost per unit flow* along the arc, C_{ij}. Each node i is assigned a real number, f_i which measures the *net outflow* at that node. If $f_i > 0$, we view node i as a *source*, if $f_i < 0$ it is a *sink*, and if $f_i = 0$, it is a *transshipment point*.

From a physical perspective, we may view the network as modeling the flow of fluid through pipes. Sources and sinks are locations from which fluid enters and exits the system, with rate of fluid entry or exit at node i measured by f_i. Arcs represent pipes, with M_{ij} denoting the maximum possible fluid flow through the pipe starting at node i and ending at node j. Figure 3.1 illustrates flow between two nodes. Note that there exist two flow capacities between these two nodes, M_{ij} and M_{ji}, and that these values need not be equal. For example, if $M_{ij} > 0$ and $M_{ji} = 0$, then flow is permitted only in the direction from node i and to j but not in the reverse direction.

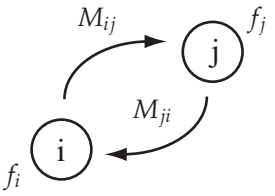


FIGURE 3.1: Two nodes in the minimum cost flow problem.

The minimum cost network flow problem is the LP that seeks to determine the flow values between nodes that minimize the total cost of fluid flow through the network, subject to the flow constraints and the assumption that conservation of flow occurs at each node. By conservation of flow we mean the combined net outflow at node i due to fluid both flowing in from and also flowing out to adjacent nodes is given by f_i. If x_{ij}, where $1 \leq i, j \leq n$, is the decision variable representing the flow from node i to node j, the minimum cost network flow LP is given by (3.4).

$$\begin{aligned} &\text{minimize } z = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij} \qquad (3.4)\\ &\text{subject to}\\ &\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = f_i, \text{ for } 1 \le i \le n\\ &x_{ij} \le M_{ij} \text{ for } 1 \le i, j \le n\\ &x_{ij} \ge 0 \text{ for } 1 \le i, j \le n. \end{aligned}$$

Note that the conservation constraints, when summed over i, lead to

$$\begin{aligned} \sum_{i=1}^{n} f_i &= \sum_{i=1}^{n}\left(\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji}\right)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ji}\\ &= 0. \end{aligned}$$

Hence, a necessary condition for (3.4) to be feasible is that $\sum_{i=1}^{n} f_i = 0$.

Both the transportation and transshipment problems from Section 3.2 are examples of minimum cost network flow LPs. For the first type of problem, there exist two classes of nodes, those at which the net outflow is positive, the supply points, and those at which it is negative, the demand points. The net outflow numbers represent the respective supply and demand amounts. The available supply from the first type of point may be used as the flow capacity to any demand point. The flow capacity along any arc starting and ending at two supply points as well as along any arc starting at a demand point is zero. For the transhipment LP, there exists a third class of nodes, namely those at which the net outflow is zero.

We already know from Theorem 3.2.1 that the solution of the balanced transportation problem having integer supply and demand amounts is itself integer-valued. Under appropriate conditions, this result generalizes to the minimum network flow problem problem and is spelled out by Theorem 3.3.1, sometimes known as the *Integrality Theorem*.

Theorem 3.3.1. Assume in LP (3.4) that the flow constraints, M_{ij}, where $1 \le i, j \le n$, and net outflows f_i, where $1 \le i \le n$ are integer-valued. Then the decision variable values in any basic feasible solution are also integer-valued.

3.3.2 Formulating and Solving the Minimum Cost Network Flow Problem with Maple

Maple list, array, and matrix structures provide a straightforward means for solving minimum network flow problems. For example, consider the network of 5 nodes depicted in Figure 3.2. Nodes are labeled I-V, together with their corresponding outflows. Along each arc is an ordered pair, whose entries represent cost and flow capacity, respectively, along the arc.

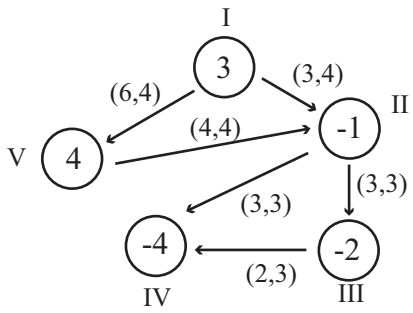


FIGURE 3.2: Five-node network with corresponding outflow numbers, costs, and flow capacities.

The following worksheet constructs the solution to the corresponding minimum network flow problem. Output has been suppressed with the exception of the final solution.

```
> with(LinearAlgebra):with(Optimization):
> f:=[3,-1,-2,-4,4]:
  # Outflow numbers corresponding to nodes.
> M:=Matrix(5,5,[0,4,0,0,4,0,0,3,3,0,0,0,0,3,0,0,0,0,0,0,0,4,0,0,0]):
  # Matrix of flow capacities.
> C:=Matrix(5,5,[0,3,0,0,6,0,0,3,3,0,0,0,0,2,0,0,0,0,0,0,0,4,0,0,0]):
  # Matrix of unit flow costs.
> x:=array(1..5,1..5):
  # Array of decision variables.
> z:=add(add(C[i,j]*x[i,j],j=1..5),i=1..5):
  # Network cost function.
> FlowConservation:=
  seq(add(x[i,j],j=1..5)-add(x[j,i],j=1..5)=f[i],i=1..5):
  # Conservation of flow constraints.
```

```
> FlowCapacities:=seq(seq(x[i,j]<=M[i,j],j=1..5),i=1..5):
  # Flow capacity constraints.
> LPSolve(z,[FlowConservation,FlowCapacities],assume=nonnegative):
  # Solve minimum cost flow LP, suppressing output.
> assign(%[2]):
  # Assign the decision variables to their solution values.
> print(x,z);
  # Print the solution array along with objective value.
```

$$x := \begin{bmatrix} 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix}, 45$$

3.3.3 The Shortest Path Problem

One of the simplest examples of a minimum cost network flow problem arises when we seek the shortest path of travel starting at one location and ending at another, given various routes from which to choose. The starting point may be viewed as a source with net outflow 1, the destination as a sink with outflow -1, and intermediate locations as transshipment points. If distances between locations are viewed as transportation costs, the shortest path problem reduces to that of a transshipment problem whose objective is to minimize the cost of transporting one unit from the source to the sink.

For example, suppose an individual wishes to travel from his home in Grand Rapids, Michigan to Graceland, Tennessee and is considering routes that pass through the cities of Chicago, Indianapolis, Lansing, and St. Louis. Approximate distances between the various pairs of cities are shown in Figure 3.3. Assume that the lack of a single segment connecting two cities, e.g., Grand Rapids and St. Louis, indicates that a direct route between those two cities is not under consideration. That arrows in the diagram are bidirectional indicates that travel is possible in either direction and that the corresponding transportation costs, i.e., distances, are equal.

Our goal is to determine the path of shortest length that starts at Grand Rapids and ends at Graceland, traveling only upon segments shown in Figure 3.3.

We begin by numbering the cities as shown in the figure and by letting x_{ij}, where $1 \le i, j \le 6$, denote the fraction of the one unit that is transported from city i to city j. Because city 1 is a source, city 6 is a sink, and all other cities are transshipment points, we have $f_1 = 1$, $f_6 = -1$, and $f_2 = f_3 = f_4 = f_5 = 0$. For pairs of cities connected by arcs, we set the flow capacity equal to 1. For pairs not connected, we set the capacity to 0. Thus, since all net outflows

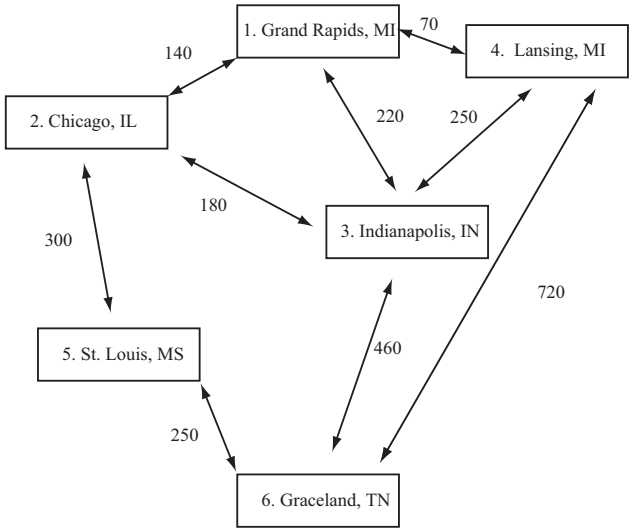


FIGURE 3.3: Driving distances between various cities.

and all flow capacities are integer-valued, we conclude by Theorem 3.3.1 that the LP's solution will yield integer-valued decision variable values. In particular, if travel is not possible between cities i and j, we know a priori that $x_{ij} = x_{ji} = 0$.

Costs associated with the objective function are determined using inter-city driving distances, which we record by the matrix

$$C = \begin{bmatrix} 0 & 140 & 220 & 70 & 0 & 0 \\ 140 & 0 & 180 & 0 & 300 & 0 \\ 220 & 180 & 0 & 250 & 0 & 460 \\ 70 & 0 & 250 & 0 & 0 & 720 \\ 0 & 300 & 0 & 0 & 0 & 250 \\ 0 & 0 & 460 & 720 & 250 & 0 \end{bmatrix}. \tag{3.5}$$

Note that our flow capacity constraints already guarantee that the decision variable corresponding to pairs of cities between which travel is not possible must equal 0 in the solution. Thus, in our cost matrix, we may set to zero (or any value, for that matter) the cost corresponding to any such pair of cities. Observe also that the matrix C is symmetric, i.e., $C = C^t$, which reflects the facts flow is bidirectional and that transportation costs between any two nodes are the same, regardless of direction.

Using C we then have the objective function

$$z = \text{Total Distance Travelled} = \sum_{i=1}^{6} \sum_{j=1}^{6} C_{ij} x_{ij}. \tag{3.6}$$

Using the objective z, along with the flow capacities, and conservation of flow constraints we have the information required to solve the minimum cost flow problem. Its solution, as computed using Maple, is given by $z = 680$, $x_{13} = x_{36} = 1$ and all other decision variables equal to zero. Thus, the shortest path corresponds to the 680-mile route from Grand Rapids to Indianapolis to Graceland.

Clearly, many variants of this problem exist. For example, the objective can become one of minimizing the time, not the distance, required to travel from the source to the sink.

Waypoint 3.3.1. Suppose that driving speeds between all pairs of cities in Figure 3.3 is 70 miles per hour, with the exception of the segment from Grand Rapids to Indianapolis, where speed is limited to 55 miles per hour. Assuming that one travels from one city to another at a constant speed equal to the legal maximum, determine the route that minimizes the time required to travel from Grand Rapids to Graceland.

3.3.4 Maximum Flow Problems

The maximum flow problem is another example of a minimum cost network flow problem, which, like the shortest path problem, involves a single source, a single sink, and transshipment points.

For example, a municipality wishes to design a plan that maximizes the flow of wastewater to a local sewage treatment plant. Figure 3.4 depicts a community (labeled I), which sends its wastewater to three different pumping stations (labeled II, III, and IV), which, in turn, transport the wastewater to a sewage treatment facility (labeled V). Directions of possible flow along pipelines are depicted by arrows, with the flow capacity of each pipe, as measured in units of volume per unit time, denoted by the number adjacent to the corresponding arrow. The city seeks to determine what flow through the given pipes will maximize total possible flow from the city and to the treatment plant, subject to the prescribed flow capacities.

We begin by numbering the city, pumping station, and treatment plant as

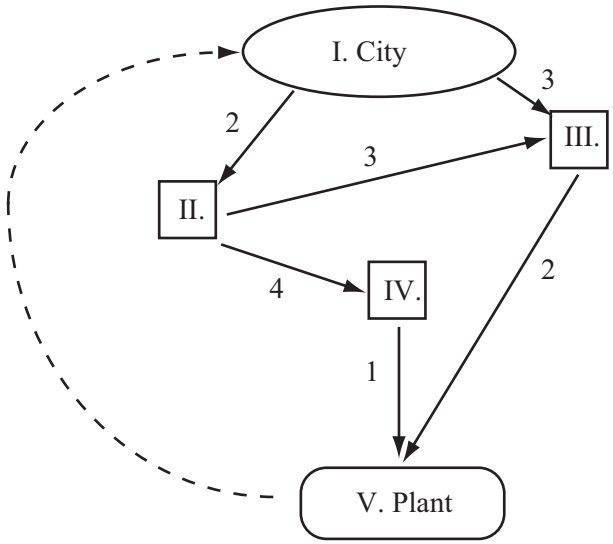


FIGURE 3.4: Wastewater flow diagram.

shown in Figure 3.4, and we let x_{ij}, where $1 \leq i, j \leq 5$, denote the flow rate (in volume of flow per unit time) from location i to location j. Because we do not know the net outflow form the city (it is, after all, the quantity we seek to maximize), we must alter the problem slightly without changing the nature of the solution. The key is to add an "artificial" arc connecting the treatment plant to the city and to view both the plant and city as transshipment points. Hence, $f_1 = f_2 = f_3 = f_4 = f_5 = 0$. We also set the flow capacity along this new arc to a large number, m, such as any number greater than the sum of the already-existing capacities. We use $m = 100$.

Our goal is to maximize the flow leaving the city or, by conservation, the flow along the "artificial" arc from the plant to the city. In other words, we seek to maximize x_{51}, equivalently minimize $z = -x_{51}$. Hence, we set $C_{51} = -1$ and all other unit flow costs to zero, which gives us $C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$.

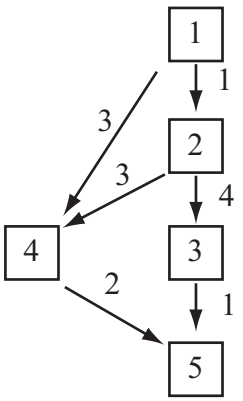
Along with $M = \begin{bmatrix} 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 \\ 100 & 0 & 0 & 0 & 0 \end{bmatrix}$, we have all the data needed to solve the problem of maximizing the flow through the city.

Waypoint 3.3.2. Use Maple to verify that the flow from the city is maximized when $x_{12} = x_{24} = x_{45} = 1$, $x_{13} = x_{35} = 2$, and $x_{51} = 3$.

Exercises Section 3.3

1. A network consists of 5 nodes, labeled 1-5, whose net outflows are $2, -3, 4, -2$ and -1, respectively. Assume that the flow capacity along each arc connecting any two pair of nodes is 2, with the exception of any arc starting at the third node, which has flow capacity 1. If the matrix
$$C = \begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 3 & 3 & 2 & 1 \\ 3 & 4 & 1 & 2 & 1 \\ 4 & 4 & 3 & 3 & 3 \\ 1 & 1 & 2 & 1 & 3 \end{bmatrix}$$
records the cost per unit flow between any two nodes, determine the solution to the corresponding minimum cost network flow problem.

2. The figure below illustrates a network consisting of 5 nodes and 6 paths, together with the flow capacity along each arc. Find the maximum flow from node 1 to node 5.



3. Consult a road atlas and determine the route of shortest driving distance from Billings to Great Falls, Montana that uses only interstate and U.S. highways. Then determine the route of shortest time duration, assuming interstate highway speeds of 70 miles per hour and U.S. highway speeds of 55 miles per hour.

4. A hospital discharges to its outpatient system a portion of its patients, each of whom receives one of three types of follow-up services: speech therapy, physical therapy, and occupational therapy. Each therapy type is performed at two different outpatient clinics.[2] Table 3.8 indicates the maximum number of new outpatient admissions of a specified therapy type that each clinic can accept during any given day.

TABLE 3.8: Maximum number of outpatient admissions of each therapy type at each clinic

Therapy Type/Clinic	Clinic 1	Clinic 2
Speech	3	2
Physical	6	4
Occupational	2	3

Assume that the hospital can discharge at most 10 patients daily to each clinic and that both speech and occupational therapy services can perform up to 5 daily discharges from the outpatient system but that physical therapy can only perform up to 4. Assume also that at most 3 patients can have services transferred from one clinic to the other on any given day.

Suppose the hospital represents the source and the state of complete discharge denotes the sink. Using the clinics and therapy types as nodes in the network, determine the maximum number of patients the hospital can discharge daily for outpatient services.

[2]Based upon Duncan and Noble, [12], (1979).

Chapter 4

Duality and Sensitivity Analysis

4.1 Duality

Duality occurs when two interrelated parts comprise the whole of something. In the context of linear programming, duality refers to the notion that every LP has a corresponding *dual* LP, whose solution provides insight into the original LP.

4.1.1 The Dual of an LP

Throughout this section we shall refer to the standard maximization problem having n decision variables and m constraints, written in matrix inequality form as

$$\begin{aligned} &\text{maximize } z = \mathbf{c} \cdot \mathbf{x} \\ &\text{subject to} \\ &\quad A\mathbf{x} \leq \mathbf{b} \\ &\quad \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{4.1}$$

where $\mathbf{x}$ belongs to $\mathbb{R}^n$, $\mathbf{c}$ is a row vector in $\mathbb{R}^n$, $\mathbf{b}$ belongs to $\mathbb{R}^m$, and A is an m-by-n matrix. The prototype example is of course the *FuelPro* LP, in which case $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 4 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix}$, and $A = \begin{bmatrix} 1 & 0 \\ 2 & 2 \\ 3 & 2 \end{bmatrix}$. The expanded form of this LP is given by:

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 \\ &\text{subject to} \\ &\quad x_1 \leq 8 \\ &\quad 2x_1 + 2x_2 \leq 28 \\ &\quad 3x_1 + 2x_2 \leq 32 \\ &\quad x_1, x_2 \geq 0. \end{aligned} \tag{4.2}$$

Let $\mathbf{y}$ denote a row vector in $\mathbb{R}^m$. The *dual LP* of (4.1) is written in matrix

inequality form as

$$\begin{aligned} &\text{minimize } w = \mathbf{y} \cdot \mathbf{b} \\ &\text{subject to} \\ &\mathbf{y}A \geq \mathbf{c} \\ &\mathbf{y} \geq \mathbf{0}. \end{aligned} \tag{4.3}$$

There are several features to observe about the dual LP. First, its goal is minimization. Second, its objective function coefficients are determined from the right-hand sides of the original LP's constraints. (We call the original LP the *primal LP*.) Finally, the constraints of the dual LP are all greater-than-or-equal-to, and the right-hand sides now are the objective coefficients of the primal LP.

In the case of the *FuelPro* model, $\mathbf{y}$ is a row vector in $\mathbb{R}^3$, which we denote as $\mathbf{y} = [y_1, y_2, y_3]$, and the dual LP's objective function is given by

$$w = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 8 \\ 28 \\ 32 \end{bmatrix} = 8y_1 + 28y_2 + 32y_3.$$

Substituting the other vector/matrix quantities from the *FuelPro* LP into (4.3), we obtained the expanded form of the dual of (4.2),

$$\begin{aligned} &\text{minimize } w = 8y_1 + 28y_2 + 32y_3 \\ &\text{subject to} \\ &y_1 + 2y_2 + 3y_3 \geq 4 \\ &2y_2 + 2y_3 \geq 3 \\ &y_1, y_2, y_3 \geq 0. \end{aligned} \tag{4.4}$$

Viewing the dual LP in the form (4.4) gives us better insight into the correspondence between primal LP and its dual. Whereas the primal LP involved three constraints in two decision variables, the dual LP involves two constraints in three decision variables. Moreover, there is a natural correspondence between each decision variable in the dual and a constraint in the primal. For example, by comparing variable coefficients, we see that the decision variable y_3 in (4.4) corresponds to the third constraint in the primal.

Before proceeding, we pause to emphasize why duality is important.

1. First, we will establish that the solution to an LP can be obtained by solving its dual. Thus, one has the freedom to solve either of the LPs, depending upon which is computationally easier. In fact, later in this chapter, we develop a variant of the simplex algorithm, known as the *dual simplex algorithm*, which capitalizes upon this result.

2. In the next section, we investigate *sensitivity analysis*. As the term suggests, it determines the extent to which optimal solutions remain stable under parameter changes in the original LP, e.g., under changes in objective coefficients, constraint bounds, etc. Duality plays a central role in this analysis.

3. Finally, there is an inherent aesthetic beauty to duality in that it provides a fresh perspective from which to view the simplex algorithm, reinforces important linear algebra concepts, and demonstrates that there is much more to linear programming than mere matrix computations.

◆ ——— ◆

Waypoint 4.1.1. Consider the LP

$$\begin{aligned} &\text{maximize } z = 3x_1 + 5x_2 + 6x_3 \\ &\quad \text{subject to} \\ &2x_1 + x_2 + x_3 \leq 1 \\ &x_1 + 2x_2 + x_3 \leq 2 \\ &x_1 + x_2 + 2x_3 \leq 4 \\ &x_1 + x_2 + x_3 \leq 3 \\ &\quad x_1, x_2, x_2 \geq 0. \end{aligned}$$

Express this LP in the matrix inequality form (4.1), clearly specifying all matrices and vectors and checking that all resulting matrix/vector products are well defined. Then express the dual LP in both matrix inequality and expanded forms. Finally, solve each LP using whatever means you find most convenient. Try to look for relationships between the solutions you obtain.

◆ ——— ◆

4.1.2 Weak and Strong Duality

The path to establishing relationships between an LP and its corresponding dual begins with the following result, which we refer to as the Weak Duality Theorem.

Theorem 4.1.1. Suppose $\mathbf{x}_0$ is primal-feasible, meaning it satisfies the constraints and sign conditions in (4.1), and $\mathbf{y}_0$ is dual-feasible, meaning it satisfies the constraints and sign restrictions in (4.3). Then

$$\mathbf{c}\mathbf{x}_0 \leq \mathbf{y}_0\mathbf{b}.$$

Proof. The proof is straightforward and as follows:

$$\begin{aligned} \mathbf{c}\mathbf{x}_0 &\leq (\mathbf{y}_0 A)\,\mathbf{x}_0 && (\mathbf{y}_0 \text{ is dual-feasible}) \\ &= \mathbf{y}_0\,(A\mathbf{x}_0) && (\text{associativity}) \\ &\leq \mathbf{y}_0\mathbf{b}. && (\mathbf{x}_0 \text{ is primal-feasible}). \end{aligned}$$

□

The consequences of the Weak Duality Theorem are significant. For example, suppose $\mathbf{x}_0$ is primal-feasible, $\mathbf{y}_0$ is dual-feasible, and $\mathbf{c}\mathbf{x}_0 = \mathbf{y}_0\mathbf{b}$. Then $\mathbf{x}_0$ and $\mathbf{y}_0$ are optimal solutions to their respective LPs since weak duality implies the primal objective value is no larger than $\mathbf{c}\mathbf{x}_0$ and the dual objective value is no smaller than $\mathbf{y}_0\mathbf{b}$.

Alternatively, suppose, that the primal LP is unbounded, meaning $\mathbf{c}\mathbf{x}_0$ can be made arbitrarily large. If the dual LP is feasible then there exists $\mathbf{y}_0$ such that $\mathbf{y}_0\mathbf{b}$ is a finite, real number. But the unboundedness of the primal guarantees the existence some $\mathbf{x}_0$ satisfying $\mathbf{c}\mathbf{x}_0 > \mathbf{y}_0\mathbf{b}$. This inequality contradicts the Weak Duality Theorem, implying that if the primal LP is unbounded, the dual LP must be infeasible.

Of particular interest is the case when both the primal and dual possess optimal solutions. Key to understanding the consequence of this situation is the result from Theorem 2.4.1 of Section 2.4.3. Recall that after each iteration of the simplex algorithm, the tableau matrix corresponding to (4.1) can be written in the form

$$\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0}_{m\times 1} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times m} & 0 \\ \mathbf{0}_{m\times 1} & A & I_m & \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{c}+\mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0}_{m\times 1} & MA & M & M\mathbf{b} \end{bmatrix}, \tag{4.5}$$

where $\mathbf{y}$ is a row vector in $\mathbb{R}^m$ and M is an m-by-m matrix.

Reintroducing the top row of variable labels to (4.5), we obtain the tableau shown in Table 4.1.

TABLE 4.1: General form of simplex tableau for LP (4.1)

z	$\mathbf{x}$	$\mathbf{s}$	RHS
1	$-\mathbf{c}+\mathbf{y}A$	$\mathbf{y}$	$\mathbf{y}\mathbf{b}$
$\mathbf{0}$	MA	M	$M\mathbf{b}$

The result given in Table 4.1 holds for any iteration of the simplex algorithm. However, the algorithm for this maximization problem terminates when all coefficients in the top row of the tableau are nonnegative, which forces

$$-\mathbf{c}+\mathbf{y}A \geq \mathbf{0} \quad \text{i.e.} \quad \mathbf{y}A \geq \mathbf{c}, \qquad \text{and} \quad \mathbf{y} \geq \mathbf{0}.$$

But by (4.3), this is precisely what it means to say that $\mathbf{y}$ is dual-feasible. In others words, at the final iteration of the simplex algorithm applied to an LP, we simultaneously obtain the optimal solution to the LP, $\mathbf{x}_0$, as well as a dual-feasible point, $\mathbf{y}_0 \stackrel{\text{def}}{=} \mathbf{y}$. Moreover, when this occurs, the optimal primal objective value, $z_0 = \mathbf{c} \cdot \mathbf{x}_0$, equals the right-most entry in the top row of Table 4.1, which is $\mathbf{y}_0\mathbf{b}$. By Theorem 4.1.1, $\mathbf{y}_0$ is an optimal solution of the dual LP with corresponding objective value

$$w_0 \stackrel{\text{def}}{=} \mathbf{y}_0 \cdot \mathbf{b} = z_0 = \mathbf{c} \cdot \mathbf{x}_0.$$

Thus we have proven the following result, which we refer to as the Strong Duality Theorem.

Theorem 4.1.2. If LP (4.1) has an optimal solution $\mathbf{x}_0$, then its dual LP (4.3) also has an optimal solution, $\mathbf{y}_0$, and their corresponding objective function values, z_0 and w_0, are equal. In other words, there is equality in Weak Duality at the optimal solution, and

$$z_0 = \mathbf{c}\mathbf{x}_0 = \mathbf{y}_0\mathbf{b} = w_0.$$

Moreover, the optimal dual decision variable values, $\mathbf{y}_0$, are given by the coefficients of the slack variables in the top row of the primal LP's final tableau.

By combining the result of Theorem 4.1.2, the fact that the dual of an LP's dual is the original LP, and arguments such as those following the proof of the Weak Duality Theorem (Theorem 4.1.1), we arrive at the following major result, which summarizes the three possible outcomes for an LP and its dual:

Theorem 4.1.3. For an LP and its dual LP, one of the three possible outcomes must occur:

1. If the primal LP has an optimal solution, then so does the dual LP and the conclusions of both the Weak and Strong Duality Theorems hold.
2. If the primal LP is unbounded, then the dual LP is infeasible.
3. If the primal LP is infeasible, then the dual LP is either infeasible or unbounded.

Let us recap the two means of obtaining the dual LP solution. One way is to solve it directly. For the *FuelPro* LP, this means using the Big M method (Section 2.3) and introducing excess and artificial variables as follows:

$$\text{minimize } w = 8y_1 + 28y_2 + 32y_3 + 100a_1 + 100a_2 \tag{4.6}$$

subject to

$$\begin{aligned} y_1 + 2y_2 + 3y_3 - e_1 + a_1 &= 4 \\ 2y_2 + 2y_3 - e_2 + a_2 &= 3 \\ y_1, y_2, y_3, e_1, e_2, a_1, a_2 &\geq 0. \end{aligned}$$

Here we use $M = 100$. Each iteration of the simplex algorithm yields a dual-feasible solution with corresponding objective value, **yb**, which decreases from one iteration to the next. The tableau obtained at the final iteration is shown in Table 4.2.

TABLE 4.2: Final tableau for *FuelPro* dual LP after being solved with the Big M Method

w	y_1	y_2	y_3	e_1	e_2	a_1	a_2	RHS
1	-4	0	0	-4	-10	-96	-90	46
0	1	0	1	-1	1	1	-1	1
0	-1	1	0	0	$-\frac{3}{2}$	-1	$\frac{3}{2}$	$\frac{1}{2}$

At the final iteration, we achieve an optimal solution of $y_1 = 0$, $y_2 = \frac{1}{2}$, and $y_3 = 1$, which agrees with the slack variable coefficients in the top row of the primal solution tableau. Furthermore, the optimal dual objective value, $w_0 = 46$, equals the optimal objective value from the primal.

The second means of arriving at the dual solution is to do so indirectly from the primal LP. At each iteration of the algorithm applied to the primal, we obtain, from the top row of the primal tableau, values for $\mathbf{y}$, $\mathbf{y}A$, and **yb**, which are listed in Table 4.3.

TABLE 4.3: Top rows of tableau for iterations of primal *FuelPro* LP (The z column has been omitted)

Iteration	$[-\mathbf{c} + \mathbf{y}A]_1$	$[-\mathbf{c} + \mathbf{y}A]_2$	y_1	y_2	y_3	**yb**
Zero	-4	-3	0	0	0	0
First	0	-3	4	0	0	32
Second	0	0	$-\frac{1}{2}$	0	$\frac{3}{2}$	44
Third	0	0	0	$\frac{1}{2}$	1	46

In this case, as the column headings show, we start with **yb** as small as sign restrictions allow (zero!) and iterate until we obtain a value of $\mathbf{y}$ that is dual-feasible. At each iteration, **yb** increases as $\mathbf{y}$ moves towards dual feasibility. Again, at the optimal solution, our dual objective function and decision variable values agree with those predicted by Theorem 4.1.2.

A close inspection of Table 4.2 reveals other interesting patterns. First, the coefficients of the excess variables in the top row of the tableau are the additive inverses of the decision variable values in the optimal solution of the primal. We will not prove this general result, but it illustrates how we can solve the original LP by solving its corresponding dual if doing so appears to require fewer computations, for example if the original LP has far more constraints than variables.

Second, recall that a constraint of an LP is binding if the slack variable corresponding to that constraint has a solution value of zero. In the case of the *FuelPro* LP, the decision variable, y_1, is zero in the dual LP's solution, and yet only the corresponding first constraint in the primal LP is non-binding. In the solution of the primal, neither decision variable is zero, yet both constraints in the dual LP are binding. These results suggest a possible relationship between solution values of one LP and the corresponding constraints in the other, which is true in general and is summarized by the following theorem, which we refer to as the *Complementary Slackness Property*.

Theorem 4.1.4. Assume $\mathbf{x}_0$ is a feasible solution to LP (4.1) and $\mathbf{y}_0$ is a feasible solution to (4.3). Then a necessary and sufficient condition for both to be simultaneously optimal solutions of their respective LPs is that

$$[\mathbf{y}_0 A - \mathbf{c}]_i [\mathbf{x}_0]_i = 0 \quad i = 1, 2, \ldots, n,$$

and

$$[\mathbf{y}_0]_j [\mathbf{b} - A\mathbf{x}_0]_j = 0 \quad j = 1, 2, \ldots, m.$$

In other words, we have obtained an optimal solution to an LP and its dual if and only if both of the following two conditions hold:

- Each decision variable in the primal is zero or the corresponding constraint in the dual is binding.
- Each decision variable in the dual is zero or the corresponding constraint in the primal is binding.

Proof. Feasibility and associativity dictate

$$\mathbf{c}\mathbf{x}_0 \leq (\mathbf{y}_0 A)\, \mathbf{x}_0 = \mathbf{y}_0\, (A\mathbf{x}_0) \leq \mathbf{y}_0 \mathbf{b}. \tag{4.7}$$

By Strong Duality, both solutions $\mathbf{x}_0$ and $\mathbf{y}_0$ are simultaneously optimal if and only if $\mathbf{c}\mathbf{x}_0 = \mathbf{y}_0\mathbf{b}$. In light of inequality (4.7), it follows that $\mathbf{x}_0$ and $\mathbf{y}_0$ are simultaneously optimal if and only if

$$0 = (\mathbf{y}_0 A)\, \mathbf{x}_0 - \mathbf{c}\mathbf{x}_0 \quad \text{and} \quad 0 = \mathbf{y}_0 \mathbf{b} - \mathbf{y}_0\, (A\mathbf{x}_0)\,,$$

which is equivalent to

$$0 = (\mathbf{y}_0 A - \mathbf{c})\, \mathbf{x}_0 \quad \text{and} \quad 0 = \mathbf{y}_0\, (\mathbf{b} - A\mathbf{x}_0)\,. \tag{4.8}$$

The constraint and sign conditions guarantee that all entries of the vectors

$$\mathbf{y}_0 A - \mathbf{c}, \quad \mathbf{x}_0, \quad \mathbf{y}_0, \text{ and } \mathbf{b} - A\mathbf{x}_0$$

are nonnegative. Thus the product of any two corresponding entries in either matrix product from (4.8) must be zero. Hence $\mathbf{x}_0$ and $\mathbf{y}_0$ are simultaneously optimal if and only if

$$[\mathbf{y}_0 A - \mathbf{c}]_i [\mathbf{x}_0]_i = 0 \quad i = 1, 2, \ldots, n,$$

and

$$[\mathbf{y}_0]_j [\mathbf{b} - A\mathbf{x}_0]_j = 0 \quad j = 1, 2, \ldots, m.$$

□

Because $\mathbf{y}_0$ is represented by the slack variable coefficients in the top row of the primal LP's final tableau, complementary slackness is an ideal tool for checking whether a feasible point of an LP is in fact an optimal solution (See Exercise 8).

4.1.3 An Economic Interpretation of Duality

Perhaps the best means of interpreting the dual of the *FuelPro Petroleum Company*LP begins with a dimensional analysis. The original problem dictates that the units of x_1 and x_2 are gallons of premium fuel and gallons of regular unleaded fuel, respectively. The corresponding components of $\mathbf{c}$ (scaled) are \$ per gallon of premium and dollars per gallon of regular unleaded. The respective components of $\mathbf{b}$ are gallons of premium, gallons of stock A, and gallons of stock B. The entries of A vary and are either dimensionless (row 1), gallons of stock A per gallon of fuel type (row 2), or gallons of stock B per gallon of fuel type (row 3). It follows that the units of the components of $\mathbf{y}$ are \$ per gallon premium, \$ per gallon of stock A, and \$ per gallon of stock B, respectively.

Here again is the dual of the *FuelPro* LP:

$$\begin{aligned} &\text{minimize } w = 8y_1 + 28y_2 + 32y_3 \\ &\quad \text{subject to} \\ y_1 + 2y_2 + 3y_3 &\geq 4 \\ 2y_2 + 2y_3 &\geq 3 \\ y_1, y_2, y_3 &\geq 0. \end{aligned} \tag{4.9}$$

Suppose that *FuelPro Petroleum Company*considers selling all its assets. These assets arise from three sources corresponding to the three original constraints:

the availability of premium and the availability of stocks A and B. The interested buyer seeks to purchase all assets (at respective prices of y_1, y_2, and y_3) and to do so at minimum cost. This goal yields the minimization objective w in (4.9).

For the dual constraints, the profit earned from each fuel grade can be viewed as coming from one of the three preceding asset types. The extent to which each type contributes to a particular fuel grade's profit is determined by an appropriate entry in A. From *FuelPro Petroleum Company*'s perspective, the values of y_1, y_2, and y_3 must guarantee that the amount earned from the sale of the three assets is worth at least as much as the profit these assets currently generate, be it profit stemming from premium or from regular unleaded. This fact motivates an interpretation of each of the two constraints in the dual LP.

4.1.4 A Final Note on the Dual of an Arbitrary LP

Throughout this section, for purposes of simplicity, we have focused on a standard maximization LP such as (4.1). However, the dual of any LP is well defined, and all results of this section remain valid. Key to formulating the general dual is recognizing that each constraint in the primal corresponds to a dual variable and that each primal objective coefficient corresponds to a dual constraint. Table 4.4 summarizes the general formulation for an LP having m constraints in n decision variables. In this table, "u.r.s." denotes a decision variable that is *unrestricted in sign*.

TABLE 4.4: Guide to the general dual formulation

LP	Dual LP
max **cx**	min **yb**
$\sum_{i=1}^{n} a_{ij}x_i \leq b_j$ for some $1 \leq j \leq m$	$y_j \geq 0$ for this same j
$\sum_{i=1}^{n} a_{ij}x_i = b_j$ for some $1 \leq j \leq m$	y_j u.r.s. for this same j
$x_i \geq 0$ for some $1 \leq i \leq n$	$\sum_{j=1}^{m} y_j a_{ij} \geq c_i$ for this same i
x_i u.r.s. for some $1 \leq i \leq n$	$\sum_{j=1}^{m} y_j a_{ij} = c_i$ for this same i

4.1.5 The Zero-Sum Matrix Game

Duality provides an intriguing perspective from which to view the mathematics underlying a topic in game theory known as *matrix games*. In this section, we explore the connections between these two areas.

Ed and Steve love to gamble. When they can't find enough friends to play Texas Hold 'Em, they instead play a *matrix game* using the following 3-by-3 *payoff matrix*:

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 2 & 4 & -1 \\ -2 & 0 & 2 \end{bmatrix}. \tag{4.10}$$

In this game, at each play, Ed picks a column and, simultaneously, Steve picks a row. The dollar amount $a_{i,j}$ in the resulting entry then goes to Ed if it is positive and to Steve if it is negative. For example, if Ed chooses column three and Steve simultaneously chooses row one, then Steve pays Ed \$2. Since whatever dollar amount Ed wins on a play is lost by Steve and vice versa, money merely changes hands. Such a contest is known as a *zero-sum matrix game*. The columns of the payoff matrix form Ed's possible *pure strategies* and the rows of the matrix form Steve's.

An important question in game theory asks whether a matrix game, such as that played by Ed and Steve, possesses a *pure strategy Nash equilibrium*. This entity consists of a pure strategy choice for each player, which we refer to as that player's *equilibrium pure strategy*, along with the players' corresponding payoffs, all of which combine to exhibit a very special property. Namely, knowing the equilibrium pure strategy of the other player, no player can benefit by deviating from his equilibrium pure strategy while the other continues to follow his own.

Pure strategy Nash equilibria candidates consist of entries of the payoff matrix. In the case of Ed and Steve's game, there are nine such possibilities. For example, consider the entry in column three, row one of (4.10). This entry does not constitute a pure strategy equilibrium, for if Steve recognizes that Ed always chooses column three, then Steve can increase his earnings by always choosing row two instead of row one. Similar reasoning, applied to the remaining eight entries of the payoff matrix, demonstrates that no pure strategy Nash equilibrium exists for this game.

In contrast to a pure strategy, whereby Ed and Steve always choose the same column or row, respectively, a *mixed strategy* arises by assigning probabilities to the various choices. For example, Ed may choose column one, column two, and column three with respective probabilities $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{6}$. Clearly there are infinitely many possible mixed strategies for each player. A *mixed strategy Nash equilibrium* consists of a mixed strategy for each player, called the

equilibrium mixed strategy, together with corresponding average earnings, all of which combine to exhibit an important property analogous to that for the pure strategy situation. That is, knowing the equilibrium mixed strategy of the other player, no single player can increase his average earnings by deviating from his equilibrium mixed strategy while the other continues to follow his own. Every zero-sum matrix game has a mixed strategy Nash equilibrium [46]. We now seek to determine this equilibrium for Ed and Steve's matrix game.

We first observe that matrix-vector products provide a convenient means for expressing each game move. In the case when Ed chooses column three and Steve chooses row one, we can associate to Ed the column vector $\begin{bmatrix}0\\0\\1\end{bmatrix}$ and to Steve the row vector $\begin{bmatrix}1 & 0 & 0\end{bmatrix}$. The payoff is then given by

$$\begin{bmatrix}1 & 0 & 0\end{bmatrix} \cdot A \cdot \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}1 & -1 & 2\\2 & 4 & -1\\-2 & 0 & 2\end{bmatrix} \cdot \begin{bmatrix}0\\0\\1\end{bmatrix}$$
$$= 2.$$

Waypoint 4.1.2. We now consider mixed strategies.
1. Suppose Ed chooses columns one, two, and three with respective probabilities, x_1, x_2, and x_3 and that Steve always chooses row one. Use A to determine a linear function, f_1, of x_1, x_2, and x_3, that represent Ed's expected winnings as a function of these three probabilities. Repeat this process and find functions, f_2 and f_3, that represent Ed's winnings as functions x_1, x_2, and x_3, when Steve instead always chooses row two or always chooses row three, respectively. Your results should establish that

$$\begin{bmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \\ f_3(x_1, x_2, x_3) \end{bmatrix} = A\mathbf{x}, \tag{4.11}$$

where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$.
2. Now suppose Steve chooses rows one, two, and three with respective probabilities, y_1, y_2, and y_3 and that Ed always chooses column one. Use A to determine a linear function, g_1, of y_1, y_2, and y_3, that represent Steve's expected winnings as a function of these three probabilities. Repeat this process and find functions, g_2 and g_3, that represent Steve's winnings as functions y_1, y_2, and y_3, when Ed instead always chooses column two or always chooses column three, respectively. Your results should establish that

$$\begin{bmatrix} g_1(y_1, y_2, y_3) & g_2(y_1, y_2, y_3) & g_3(y_1, y_2, y_3) \end{bmatrix} = \mathbf{y}A, \tag{4.12}$$

where $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$.

If z denotes the minimum of the three columns of $A\mathbf{x}$, then $A\mathbf{x} \geq z\mathbf{e}$, where $\mathbf{e} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. Since Ed's payoffs are associated with positive entries of A, his goal is to choose x_1, x_2, and x_3 so as to maximize z. Thus he seeks to determine the solution to the LP

$$\begin{aligned} &\text{maximize } z \text{ subject to} \\ &A\mathbf{x} \geq z\mathbf{e} \\ &\mathbf{e}^t \cdot \mathbf{x} = 1 \\ &\mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{4.13}$$

The constraint $\mathbf{e}^t \cdot \mathbf{x} = 1$ reflects the requirement that $x_1 + x_2 + x_3 = 1$. It can of course be eliminated if x_3 is replaced throughout the LP by $1 - x_1 - x_2$.

Since z represents the minimum of the three columns of $A\mathbf{x}$ and Ed seeks to maximize this quantity, we say that Ed's goal is to compute a *maximin*. The solution vector, $\mathbf{x}_0$, of (4.13) constitutes Ed's equilibrium mixed strategy.

By modifying this line of reasoning, we can establish that because Steve's payoffs are associated with negative entries of A, his goal is to choose an equilibrium mixed strategy, $\mathbf{y}_0$, which forms the solution of the dual LP of (4.13). In essence, Steve's goal is to compute a *minimax*.

◆ ——— ◆

Waypoint 4.1.3. Verify that the dual LP of (4.13) can be expressed as follows:

$$\begin{aligned} &\text{minimize } w \text{ subject to} \\ &\mathbf{y}A \leq w\mathbf{e}^t \\ &\mathbf{y} \cdot \mathbf{e} = 1 \\ &\mathbf{y} \geq \mathbf{0}, \end{aligned} \tag{4.14}$$

where $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$. Then solve both (4.13) and (4.14). Your results should indicate that

$$\mathbf{x}_0 = \begin{bmatrix} 3/28 \\ 9/28 \\ 4/7 \end{bmatrix}, \mathbf{y}_0 = \begin{bmatrix} 1/2 & 5/14 & 1/7 \end{bmatrix}, \text{ and } z_0 = w_0 = \frac{13}{14}.$$

◆ ——— ◆

These results indicate that Ed's mixed strategy equilibrium is given $\mathbf{x}_0 = \begin{bmatrix} 3/28 \\ 9/28 \\ 4/7 \end{bmatrix}$, meaning he should choose column one with probability $\frac{3}{28}$, column two with probability $\frac{9}{28}$, and column three with probability $\frac{4}{7}$. Similarly, Steve's equilibrium mixed strategy of $\mathbf{y}_0 = \begin{bmatrix} 1/2 & 5/14 & 1/7 \end{bmatrix}$ dictates he should choose rows one through three with respective probabilities, $\frac{1}{2}$, $\frac{5}{14}$, and $\frac{1}{7}$.

The corresponding objective values must be equal by Theorem 4.1.2. The common value of $\frac{13}{14}$ we refer to as the *game value*. If both players follow their equilibrium mixed strategies, then, on the average, Ed will win this amount because we associate positive payoffs to him. Likewise, Steve, to whom we associate negative payoffs, will lose this amount. In this situation, a positive

game value indicates the game is biased in Ed's favor. Of course, by modifying entries of the payoff matrix A, we can create a situation in which the game value is negative, signifying a game biased in Steve's favor. A fair game is one for which the game value is zero.

We now use duality to formally verify that the equilibrium mixed strategies, $\mathbf{x}_0 = \begin{bmatrix} 3/28 \\ 9/28 \\ 4/7 \end{bmatrix}$, $\mathbf{y}_0 = \begin{bmatrix} 1/2 & 5/14 & 1/7 \end{bmatrix}$ and corresponding payoff $z_0 = w_0 = \dfrac{13}{14}$, constitute a mixed strategy Nash equilibrium for this matrix game.

Let us suppose that Ed deviates from his equilibrium strategy, choosing a mixed strategy, $\mathbf{x} \neq \mathbf{x}_0$, with corresponding earnings, z, and that Steve continues to follow his equilibrium mixed strategy, $\mathbf{y}_0$. Then

$$\begin{aligned} z &= z\mathbf{y}_0\mathbf{e} \\ &= \mathbf{y}_0(z\mathbf{e}) \\ &\leq \mathbf{y}_0(A\mathbf{x}) && \text{(since } \mathbf{x} \text{ is primal-feasible)} \\ &= (\mathbf{y}_0 A)\mathbf{x} \\ &\leq (w_0\mathbf{e}^t)\mathbf{x} && \text{(since } \mathbf{y}_0 \text{ is dual-feasible)} \\ &= w_0(\mathbf{e}^t\mathbf{x}) \\ &= z_0 && \text{(since } w_0 = z_0\text{)} \end{aligned} \tag{4.15}$$

Thus, Ed's earnings are no more than z_0. Similar reasoning demonstrates that if Steve deviates from his equilibrium strategy, choosing a mixed strategy, $\mathbf{y} \neq \mathbf{y}_0$, with corresponding earnings, w, and if Ed continues to follow his equilibrium mixed strategy, $\mathbf{x}_0$, then $w \geq w_0$. In other words, Steve does not decrease his losses.

In Section 6.4.5 we will re-examine the zero-sum matrix game from the perspective of nonlinear programming, and in Section 8.4.4 we investigate the consequences of the players having two different payoff matrices.

Exercises Section 4.1

1. In light of the discussion in this section, explain why it is appropriate to say that the goal of the simplex algorithm is to solve a feasible LP by means of determining a feasible solution to its dual.

2. Consider the LP

$$\begin{aligned} \text{maximize } z &= 3x_1 + 4x_2 \\ \text{subject to} & \\ x_1 &\leq 4 \\ x_1 + 3x_2 &\leq 15 \\ -x_1 + 2x_2 &\geq 5 \\ x_1 - x_2 &\geq 9 \\ x_1 + x_2 &= 6 \\ x_1, x_2 &\geq 0. \end{aligned}$$

By rewriting the fifth constraint as a combination of two inequalities, express this LP in the matrix inequality form (4.1). Then construct the corresponding dual LP and demonstrate explicitly, without appealing to Table 4.4, how it possesses a variable that is unrestricted in sign.

3. Prove that the dual of the dual of LP (4.1) is the original LP. (Hint: Transpose properties may prove useful.)

4. Simplex algorithm iterations applied to an LP result in the tableau (4.5).

TABLE 4.5: Tableau for Exercise 4

z	x_1	x_2	x_3	x_4	s_1	s_2	s_3	RHS
1	0	0	0	8	-7	13	9	20
0	1	0	0	7	-3	6	15	12
0	0	0	1	10	0	4	2	14
0	0	1	0	4	-2	13	7	18

   (a) How many constraints and decision variables are in the dual of this LP?

   (b) If the basic variables in the tableau are given by $\{x_1, x_2, x_3\}$, what information does Weak Duality provide about the objective value in the solution of the dual LP?

5. Consider the LP

$$\begin{aligned} \text{maximize } z &= 2x_1 - x_2 \\ \text{subject to} & \\ x_1 - x_2 &\leq 1 \\ x_1 - x_2 &\geq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

Show that both this LP and its corresponding dual are infeasible.

6. An LP has as its objective, maximization of the expression $z = 2x_1 + 3x_2$. Simplex algorithm iterations applied to the LP result in Table (4.6).

TABLE 4.6: Tableau for Exercise 6

z	x_1	x_2	s_1	s_2	s_3	RHS
1	■	■	1	0	0	■
0	$-\frac{2}{3}$	1	$\frac{1}{3}$	0	0	6
0	3	0	$-\frac{1}{2}$	1	0	3
0	$\frac{14}{3}$	0	$-\frac{1}{3}$	0	1	10

   (a) What are the constraints of the LP? (Hint: Use the matrix, M^{-1}, where M is as given in Theorem 2.4.1.)

   (b) Fill in the entries marked ■ in the tableau.

   (c) Determine the solutions of both the LP and its corresponding dual.

7. Solve the LP

$$\begin{aligned} \text{maximize } z &= x_1 + 5x_2 \\ \text{subject to} & \\ -x_1 + x_2 &\leq 4 \\ x_2 &\leq 6 \\ 2x_1 + 3x_2 &\leq 33 \\ 2x_1 + x_2 &\leq 24 \\ x_1, x_2 &\geq 0, \end{aligned}$$

   by first formulating and then solving its corresponding dual LP.

8. Consider an LP in the form of (4.1), in which

$$\mathbf{c} = \begin{bmatrix} 5 & 1 & 1 & 4 & 1 & 2 \end{bmatrix}, A = \begin{bmatrix} 1 & 2 & 4 & 3 & 4 & 6 \\ 7 & -5 & 2 & 1 & 3 & 3 \\ 4 & 5 & -2 & 1 & 6 & -2 \\ 7 & 8 & -3 & 5 & 2 & 4 \end{bmatrix}, \text{ and } \mathbf{b} = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 3 \end{bmatrix}.$$

   Suppose that

$$\mathbf{x}_0 = \begin{bmatrix} \frac{8}{17} \\ 0 \\ \frac{5}{17} \\ \frac{2}{17} \\ 0 \\ 0 \end{bmatrix}.$$

   Without performing the simplex algorithm, show that $\mathbf{x}_0$ is the optimal

solution of the LP and determine the corresponding dual solution. (Hint: Let $\mathbf{y}_0 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix}$. Calculate both $\mathbf{y}_0 A - \mathbf{c}$ and $\mathbf{b} - A\mathbf{x}_0$ and apply Theorem 4.1.4 to construct a system of equations in the variables y_1, y_2, y_3, and y_4.)

9. Determine the mixed strategy Nash equilibrium for a two-player, zero-sum matrix game having payoff matrix,

$$A = \begin{bmatrix} 2 & -3 \\ -1 & 4 \end{bmatrix}.$$

10. Suppose $\mathbf{c}$, $\mathbf{b}$, and A are as given in (4.1). Define the matrix

$$M = \begin{bmatrix} 0_{m\times m} & -A & \mathbf{b} \\ A^t & 0_{m\times n} & -\mathbf{c}^t \\ -\mathbf{b}^t & \mathbf{c} & 0 \end{bmatrix}.$$

    (a) Verify that M is a square matrix having $m + n + 1$ rows. Then show that $M^t = -M$. (A square matrix, whose transpose equals its additive inverse, is said to be *skew-symmetric*.)

    (b) Suppose that both (4.1) and (4.3) are feasible and have optimal solutions, $\mathbf{x}_0$ and $\mathbf{y}_0$. Show that the matrix inequality $M\mathbf{w} \geq \mathbf{0}$ has a solution, $\mathbf{w} = \mathbf{w}_0$, for some $\mathbf{w}_0$ belonging to $\mathbb{R}^{m+n+1}$ and satisfying $\mathbf{w}_0 \geq \mathbf{0}$ and $[\mathbf{w}_0]_{m+n+1} > 0$.

    (c) Now assume that the matrix inequality $M\mathbf{w}_0 \geq \mathbf{0}$ holds for some vector $\mathbf{w}_0$ belonging to $\mathbb{R}^{m+n+1}$ and satisfying both $\mathbf{w}_0 \geq \mathbf{0}$ and $[\mathbf{w}_0]_{m+n+1} > 0$. Show that both (4.1) and (4.3) are feasible and have optimal solutions, $\mathbf{x}_0$ and $\mathbf{y}_0$, that can be expressed in terms of $\mathbf{w}_0$. (Hint: Let $\mathbf{y}$ be the row vector in $\mathbb{R}^m$ formed using the first m components of $\mathbf{w}_0$, and let $\mathbf{x}$ be formed using the next n components, $[\mathbf{w}_0]_{m+1}, [\mathbf{w}_0]_{m+2}, \ldots, [\mathbf{w}_0]_{m+n}$. If $\kappa = [\mathbf{w}_0]_{m+n+1}$, which is positive, define $\mathbf{x}_0 = \dfrac{1}{\kappa}\mathbf{x}$ and $\mathbf{y}_0 = \dfrac{1}{\kappa}\mathbf{y}$.)

4.2 Sensitivity Analysis

Discussion of LPs thus far has focused primarily on solving them. Sensitivity analysis takes place after an LP has been solved and seeks to determine the extent to which changing model parameters affects the solution.

Recall the *FuelPro Petroleum Company*LP, given by

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 \\ &\text{subject to} \\ &x_1 \leq 8 \\ &2x_1 + 2x_2 \leq 28 \\ &3x_1 + 2x_2 \leq 32 \\ &x_1, x_2 \geq 0, \end{aligned} \tag{4.16}$$

whose final tableau is provided in Table 4.7.

TABLE 4.7: *FuelPro Petroleum Company*final tableau, (BV $= \{x_1, x_2, s_1\}$)

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	0	$\frac{1}{2}$	1	46
0	1	0	0	-1	1	4
0	0	0	1	1	-1	4
0	0	1	0	$\frac{3}{2}$	-1	10

The feasible region and the contour corresponding to the optimal solution $z = 46$ are shown in Figure 4.1.

There are many ways in which the original LP can be modified. Examples include the introduction of an additional variable, e.g., fuel type, a change in an objective coefficient, an increase or decrease in a constraint bound, or a modification in the amount of particular stock required to produce a given gallon of fuel type.

There is an important initial observation to make before undertaking a study of sensitivity analysis. Changes in the model may preserve the set of basic variables in the solution yet change their actual values and, hence, possibly change the objective function value as well. For example, Figure 4.1 illustrates how the second and third constraints of (4.16) are binding at the optimal solution, $(x_1, x_2) = (4, 10)$. Sufficiently small changes in the right-hand sides of either of these constraints will change the actual values of the current optimal solution. But these alterations will not change the fact that in the

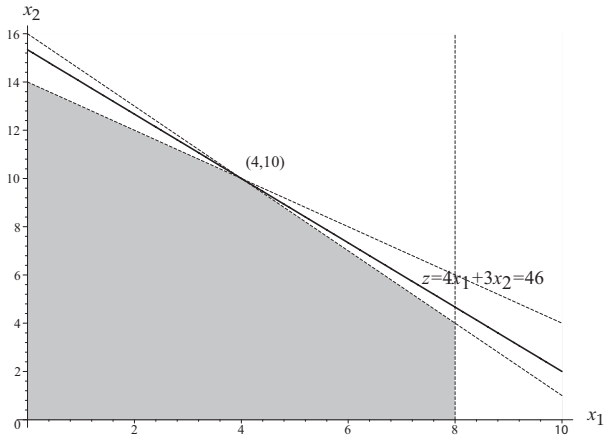


FIGURE 4.1: Feasible region for *FuelPro* LP along with contour $z = 46$.

optimal solution, the basic variables are x_1, s_1, and x_2. On the other hand, the amount of available premium fuel can decrease by up to four units and increase as large as one likes without affecting the actual values of the current optimal solution $(x_1, x_2) = (4, 10)$. This is a very different type of outcome, one that we wish to account for in our analysis.

Ideally, we wish to conduct sensitivity analysis in as efficient a manner as possible without actually resolving the LP. Crucial for doing so is the result of Theorem 2.4.1 from Section 2.4. It states that if the matrix corresponding to the initial tableau of a standard maximization LP is given by

$$\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix} \tag{4.17}$$

and if the LP is feasible and bounded, then the tableau matrix after the final iteration can be written as

$$\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0}_{1\times m} & 0 \\ \mathbf{0} & A & I_3 & \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{c}+\mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0} & MA & M & M\mathbf{b} \end{bmatrix}, \tag{4.18}$$

for some suitable 1 by m vector $\mathbf{y}$ (the solution to the dual!) and some m-by-m matrix M. In particular, all steps of the simplex algorithm used to achieve the final tableau are completely determined by the entries of $\mathbf{y}$ and M.

4.2.1 Sensitivity to an Objective Coefficient

Suppose the objective coefficients in our initial LP are represented by the vector $\mathbf{c}$ and we wish to change them by some vector quantity $\mathbf{c}_\delta$. Then the new tableau matrix becomes

$$\begin{bmatrix} 1 & -\mathbf{c} - \mathbf{c}_\delta & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix}. \tag{4.19}$$

If we apply to (4.19) the same steps of the simplex algorithm as we did to obtain (4.18), then we have

$$\begin{aligned} &\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} - \mathbf{c}_\delta & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix} \\ &\quad = \begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0} & M \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix} + \begin{bmatrix} 0 & -\mathbf{c}_\delta & \mathbf{0} & 0 \\ \mathbf{0} & 0_{m\times n} & 0_{m\times m} & \mathbf{0} \end{bmatrix} \right) \\ &\quad = \begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0} & MA & M & M\mathbf{b} \end{bmatrix} + \begin{bmatrix} 0 & -\mathbf{c}_\delta & \mathbf{0} & 0 \\ \mathbf{0} & 0_{m\times n} & 0_{m\times m} & \mathbf{0} \end{bmatrix} \\ &\quad = \begin{bmatrix} 1 & -\mathbf{c} - \mathbf{c}_\delta + \mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0} & MA & M & M\mathbf{b} \end{bmatrix}. \end{aligned} \tag{4.20}$$

By comparing (4.20) to (4.18), we see that the only changes that result from applying the same steps of the algorithm to the new LP occur in the top row of the tableau. In particular, the tableau corresponding to (4.20) reflects the need for additional iterations only if at least one entry of $-\mathbf{c} - \mathbf{c}_\delta + \mathbf{y}A$ is negative.

The *FuelPro* example illustrates an application of this principle. Suppose that the premium profit increases from 4 to $4+\delta$. By (4.20) and Table 4.7, we obtain the tableau shown in Table 4.8.

TABLE 4.8: *Fuelpro* tableau under changed premium cost

z	x_1	x_2	s_1	s_2	s_3	RHS
1	$-\delta$	0	0	$\frac{1}{2}$	1	46
0	1	0	0	-1	1	4
0	0	0	1	1	-1	4
0	0	1	0	$\frac{3}{2}$	-1	10

Recall that in the original LP, the basic variables at the final iteration were given by x_1, x_2, and s_1. To update the values of x_1 and the objective, we must pivot on the highlighted entry in Table 4.8. Doing so yields the tableau in Table 4.9.

From this tableau, we see that an additional iteration of the simplex algorithm is required only if one of $\frac{1}{2} - \delta$ or $1 + \delta$ is strictly negative. Hence if $-1 \le \delta \le \frac{1}{2}$ (meaning the premium cost stays between 3 and 4.5), then the values of the

TABLE 4.9: *Fuelpro* tableau under changed premium cost and after additional pivot

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	0	$\frac{1}{2}-\delta$	$1+\delta$	$46+4\delta$
0	1	0	0	-1	1	4
0	0	0	1	1	-1	4
0	0	1	0	$\frac{3}{2}$	-1	10

decision variables remain at $(x_1, x_2) = (4, 10)$. However, the corresponding objective function changes to $z = 46 + 4\delta$.

A slightly different type of example is given by the three-variable LP

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 + 6x_3 \\ &\text{subject to} \\ &3x_1 + x_2 + 3x_3 \leq 30 \\ &2x_1 + 2x_2 + 3x_3 \leq 40 \\ &x_1, x_2, x_3 \geq 0. \end{aligned} \tag{4.21}$$

The final tableau corresponding to this LP is given in Table 4.10.

TABLE 4.10: Final tableau for three-variable LP (4.21)

z	x_1	x_2	x_3	s_1	s_2	RHS
1	1	0	0	1	1	70
0	$\frac{4}{3}$	0	1	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{20}{3}$
0	-1	1	0	-1	1	10

The basic variables at the final iteration are given by $x_2 = 10$ and $x_3 = \dfrac{20}{3}$. Suppose in this case that the first objective coefficient increases from 4 to $4+\delta$. Then applying to the new LP the same steps of the simplex algorithm that led to Table 4.10, we obtain the tableau given in Table 4.11.

TABLE 4.11: Final tableau for modification of three-variable LP (4.21)

z	x_1	x_2	x_3	s_1	s_2	RHS
1	$1-\delta$	0	0	1	1	70
0	$\frac{4}{3}$	0	1	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{20}{3}$
0	-1	1	0	-1	1	10

The current solution remains optimal provided $\delta < 1$. In this case, no addi-

tional pivots are necessary in order to update the objective value. This stems from the fact that we elected to increase the objective coefficient of a decision variable that was nonbasic in the optimal solution of the original LP. In the case of the *FuelPro* LP, both decision variables were basic in the optimal solution of the original LP.

Duality provides an additional means of handling this situation in which the objective coefficient of a nonbasic variable is permitted to increase. From Table 4.11, we see that the solution of the dual of (**??**) is given by $\mathbf{y} = [1, 1]$. To say that the current solution of (4.21) remains optimal is to say that $\mathbf{y} = [1, 1]$ remains a feasible solution of the corresponding dual LP. In other words,

$$\mathbf{y}A = \mathbf{y}\begin{bmatrix} 3 & 1 & 3 \\ 2 & 2 & 3 \end{bmatrix} = [5, 3, 6] \geq [4 + \delta, 3, 6], \tag{4.22}$$

which implies $\delta < 1$.

In summary, when performing a sensitivity analysis that focuses on the effect of changing a coefficient in the objective function by some value δ, we start with (4.20).

1. If the coefficient corresponds to a nonbasic variable in the solution of the original LP, then the range of values δ for which the optimal solution values remain unchanged is straightforward to determine. We can subtract δ from the coefficient of the corresponding variable in the top row of the original final tableau or use duality.

2. If the coefficient corresponds to a basic variable, we should pivot to update the value of the objective function and to determine whether another iteration of the simplex algorithm is necessary. For values of δ not leading to an additional iteration, the decision variable values in the optimal solution are the same as in the original LP. However, the objective function value will now depend upon δ, in a manner determined by the pivots.

Waypoint 4.2.1. Solve the LP

$$\begin{aligned} \text{maximize } z &= 4x_1 + x_2 + 5x_3 \qquad (4.23)\\ \text{subject to} & \\ 2x_1 + x_2 + 3x_3 &\leq 14 \\ 6x_1 + 3x_2 + 3x_3 &\leq 22 \\ 2x_1 + 3x_2 &\leq 14 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

Then perform a sensitivity analysis on each of the objective function coefficients.

4.2.2 Sensitivity to Constraint Bounds

We now consider the effect of changing one or more constants that bound the right-hand sides of an LP's constraints. In terms of the matrix form (4.17), we are concerned with the outcome of increasing or decreasing one or more entries of $\mathbf{b}$. Initially, our focus will be on perturbing a single entry of the "constraint vector" $\mathbf{b}$ in (4.17), one that corresponds to a binding constraint.

Suppose we wish to increase the right-hand side of such a constraint by δ units, where $\delta < 0$ is interpreted as a decrease. Thus, the new constraint vector may be denoted as

$$\mathbf{b} + \delta\mathbf{u}.$$

Here $\mathbf{u}$ is an m-by-1 column vector consisting of all zeros, except in the entry corresponding to the constraint being changed, where a 1 is present instead.

The partitioned matrix form corresponding to the initial tableau is given by

$$\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} + \delta\mathbf{u}. \end{bmatrix} \qquad (4.24)$$

Applying the same steps of the simplex algorithm that were used to solve the

original LP, we have

$$\begin{aligned}
&\begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0} & M \end{bmatrix} \cdot \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} + \delta\mathbf{u} \end{bmatrix} \\
&\qquad = \begin{bmatrix} 1 & \mathbf{y} \\ \mathbf{0} & M \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & 0_{m\times n} & 0_{m\times m} & \delta\mathbf{u} \end{bmatrix} \right) \\
&\qquad = \begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0} & MA & M & M\mathbf{b} \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{0} & \mathbf{0} & \mathbf{y}\delta\mathbf{u} \\ \mathbf{0} & 0_{m\times n} & 0_{m\times m} & M\delta\mathbf{u} \end{bmatrix} \\
&\qquad = \begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}\,(\mathbf{b} + \delta\mathbf{u}) \\ \mathbf{0} & MA & M & M\mathbf{b} + M\delta\mathbf{u} \end{bmatrix}. \qquad (4.25)
\end{aligned}$$

Note that the top row of this matrix, with the exception of the rightmost entry, is identical to that obtained applying the simplex algorithm to the original, unmodified LP. Thus, on first inspection, (4.25) corresponds to the final tableau of the modified LP, and the set of basic variables is the same as in the original LP. However, care must be taken to ensure that all basic variables are nonnegative.

Let BV_j, where $1 \le j \le m$ denote the basic variable corresponding to row j of the final tableau. Then

$$\begin{aligned}
\mathrm{BV}_j &= [M\mathbf{b} + M\delta\mathbf{u}]_j \qquad (4.26) \\
&= [M\mathbf{b}]_j + [M\delta\mathbf{u}]_j \\
&= [M\mathbf{b}]_j + \delta\,[M\mathbf{u}]_j\,.
\end{aligned}$$

Each basic variable is currently positive in the original LP (when $\delta = 0$), unless the LP is degenerate, a case we will omit from discussion. By setting positive each of the m expressions comprising the right-hand side of (4.26) and solving for δ, we can determine the maximum and minimum allowable range of δ values that result in positive basic variable values. In general this range can be computed as

$$\max_{j=1,2,\ldots,m} \left\{ -\frac{[M\mathbf{b}]_j}{[M\mathbf{u}]_j} \;\middle|\; [M\mathbf{u}]_j > 0 \right\} < \delta < \min_{j=1,2,\ldots,m} \left\{ -\frac{[M\mathbf{b}]_j}{[M\mathbf{u}]_j} \;\middle|\; [M\mathbf{u}]_j < 0 \right\}. \qquad (4.27)$$

For example, suppose that in the *FuelPro* LP, an additional δ units of stock A becomes available, in which case the right-hand side of the second constraint in (4.16) increases from 28 to $28 + \delta$. The final tableau for the original LP is given in Table 4.12.

In this situation, the basic variables corresponding to rows 1-3 are given by

TABLE 4.12: Final tableau for original *FuelPro* LP

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	0	$\frac{1}{2}$	1	46
0	1	0	0	-1	1	4
0	0	0	1	1	-1	4
0	0	1	0	$\frac{3}{2}$	-1	10

x_1, s_1, and x_2, respectively. Since $M = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 1 & -1 \\ 0 & \frac{3}{2} & -1 \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, an increase in the right-hand side of the second constraint by δ yields

$$\begin{bmatrix} x_1 \\ s_1 \\ x_2 \end{bmatrix} = M\mathbf{b} + \delta M\mathbf{u}$$
$$= \begin{bmatrix} 4-\delta \\ 4+\delta \\ 10+\frac{3}{2}\delta \end{bmatrix}.$$

For basic variables x_1, s_1, and x_2 to all remain positive, it must be the case that $-4 < \delta < 4$, or, in other words, the amount of available stock A stays between 24 and 32 gallons. For values of δ in this range, the corresponding objective value is given by

$$z_0 = \mathbf{y}(\mathbf{b} + \delta\mathbf{u}) = 46 + \frac{1}{2}\delta.$$

A close inspection of the preceding example, together with the general objective formula $z = \mathbf{y}(\mathbf{b} + \delta\mathbf{u})$ in (4.25), illustrates a fundamental role played by slack variables in sensitivity analysis. The increase in the objective value in this particular example is

$$\frac{1}{2}\delta = y_2\delta = \mathbf{y}\delta\mathbf{u}.$$

Thus, slack variable coefficient values in the top row of the final tableau, which are represented by $\mathbf{y}$, can be used to determine a change in objective value when a constraint bound is changed. This outcome is true even when the previously discussed sensitivity analysis is not performed, provided the set of basic variables remains intact under such a change. This gives slack variables an extremely useful role in sensitivity analysis and leads us to the following definition.

Definition 4.2.1. Suppose that the optimal solution to a standard maximization LP is given by

$$z_0 = \mathbf{yb} = \sum_{j=1}^{m} y_j b_j.$$

Then the instantaneous rate of change, $\frac{\partial z_0}{\partial b_j} = y_j$, the jth slack variable coefficient value in the top row of the LP's final tableau, is called the *shadow price* of the objective function z with respect to constraint j. For a small amount of change in b_j, small enough so that the set of basic variables in the solution of the new LP is identical to that in the solution of the original LP, this shadow price can be used to compute the amount of increase or decrease in the objective function.

For example, consider a maximization LP in two decision variables whose final tableau is given by Table 4.13:

TABLE 4.13: Final tableau for sample maximization LP

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	3	0	2	20

Since $y_j \neq 0$ for $j = 1$ and $j = 3$, the Complementary Slackness Property (Theorem 4.1.4) dictates that the first and third constraints of the LP are binding and that the shadow price of z with respect to the first constraint is

$$\frac{\partial z_0}{\partial b_1} = y_1 = 3.$$

Assume that increasing the right-hand side of the first constraint by .01 does not change the set of basic variables in the new solution from that in the solution of the original LP. Then the objective z increases by $\Delta z = 3(.01) = .03$.

In summary, when performing a sensitivity analysis that focuses on the effect of changing the right-hand side of a binding constraint in a standard maximization problem, we do the following:

1. To find the permissible range of increase or decrease in the right-hand side of a constraint, we use formula 4.27 and/or the derivation that led to it. This formula provides a means for determining the updated basic variable and objective values.

2. To determine the magnitude of change solely in the objective function value caused by the change in a constraint bound, we use the shadow

price of the corresponding constraint. This is given by the corresponding slack variable coefficient in the top row of the LP's final tableau. The shadow price must be used with caution, however, as its ability to predict change in the objective value assumes that the set of basic variables remains unchanged from that in the solution of the original LP.

◆ ◆

Waypoint 4.2.2. Consider the three-variable LP from Waypoint 4.2.1.

1. Suppose that the right-hand side of a particular constraint increases by Δb and that the set of basic variables in the solution of the new LP is identical to that obtained in the original LP. Use the definition of shadow price to determine how much the objective function will increase or decrease. Perform this analysis for each constraint that is binding in the original LP.
2. By how much may the right-hand side of each binding constraint change and the set of basic variables in the solution to the resulting LP equal that from the solution to the original LP? For this range, what are the corresponding values of the basic variables and the objective function?

◆ ◆

In the preceding example, we modified the right-hand side of only one constraint. Yet the underlying principles can be easily modified to situations involving more than one constraint bound. In this case, the constraint vector $\mathbf{b}$ is adjusted by the vector $\boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix}$, where δ_i, for $i = 1, 2, 3$, denotes the change in the bound on constraint i. Now the final matrix corresponding to that in (4.25) becomes

$$\begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}(\mathbf{b} + \boldsymbol{\delta}) \\ \mathbf{0} & MA & M & M\mathbf{b} + M\boldsymbol{\delta} \end{bmatrix}. \tag{4.28}$$

For example, in the *FuelPro* LP, if the available premium increases by δ_1 and

the available stock A increases by δ_2, then $\boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ 0 \end{bmatrix}$ so that

$$\begin{bmatrix} x_1 \\ s_1 \\ x_2 \end{bmatrix} = M\mathbf{b} + M\boldsymbol{\delta} \tag{4.29}$$

$$= \begin{bmatrix} 4 - \delta_2 \\ 4 + \delta_1 + \delta_2 \\ 10 + \frac{3}{2}\delta_2 \end{bmatrix}. \tag{4.30}$$

The set of ordered pairs, (δ_1, δ_2), for which x_1, s_1, and s_2 are all nonnegative is shown in Figure 4.2. For ordered pairs in this shaded region, the new objective value is given by $z = \mathbf{y}(\mathbf{b} + \boldsymbol{\delta})$.

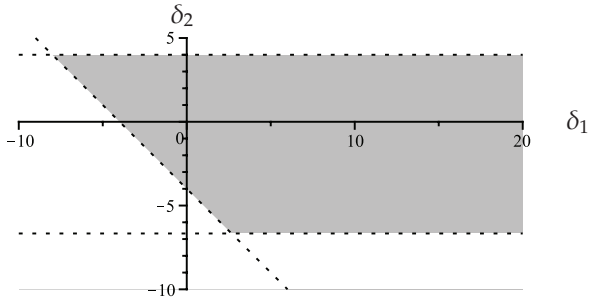


FIGURE 4.2: Ordered pairs (δ_1, δ_2) for which changes in the first two constraints of *FuelPro* LP leave the basic variables, $\{x_1, s_1, x_2\}$, unchanged.

4.2.3 Sensitivity to Entries in the Coefficient Matrix A

Performing sensitivity analysis when entries of the coefficient matrix A are changed can be complicated and tedious for reasons to be outlined momentarily. One situation in which the analysis is rather straightforward occurs when the matrix entry corresponds to a decision variable that is nonbasic in the solution of the original LP. Suppose first that we add some matrix A_δ to A, where the only nonzero entries of A_δ are those corresponding to such nonbasic variables. We leave it as an exercise to show that by applying the same elementary row operations that resulted in the final tableau for the original LP, we obtain the matrix

$$\begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}(A + A_\delta) & \mathbf{y} & \mathbf{yb} \\ \mathbf{0} & M(A + A_\delta) & M & M\mathbf{b} \end{bmatrix}. \tag{4.31}$$

In Section 4.2.2, we considered sensitivity of an LP to a coefficient in its

objective function. We observed that when we applied the same simplex algorithm steps that were used to solve the original LP, we obtained a tableau matrix whose decision variable coefficients were represented by the vector $-\mathbf{c} - \mathbf{c}_\delta + \mathbf{y}A$. We also observed that duality could be used to determine the permissible range of δ values for which the optimal solution remained unchanged. Here, the use of duality is crucial.

The reason for this fact can be seen by contrasting the two vectors $-\mathbf{c} - \mathbf{c}_\delta + \mathbf{y}A$ and $-\mathbf{c} + \mathbf{y}(A + A_\delta)$ which appear in the top rows of the respective tableau matrices (4.20) and (4.31). Changing an objective coefficient of a single decision variable in an LP only changes the coefficient of the same variable in the top row when the simplex algorithm is performed. But when an entry of A is modified, the presence of the vector-matrix product $\mathbf{y}(A + A_\delta)$ in the top row of (4.31) introduces the possibility that more than one coefficient in the top row of the tableau is altered.

Duality provides a convenient means of addressing this problem. If the vector $\mathbf{y}$ in (4.31), which is nonnegative, satisfies

$$\mathbf{y}(A + A_\delta) \geq \mathbf{c},$$

then the solution to the LP whose final tableau matrix is given in (4.31) remains optimal. Moreover, no additional pivots are needed to update any of the basic variable values. This second fact follows from noting that entries of $-\mathbf{c} + \mathbf{y}A$ corresponding to basic variables are all zero (in the solution of the original LP), as are those in $\mathbf{y}A_\delta$. (Recall in our choice of A_δ that column entries of A_δ corresponding to basic variables must equal zero.)

For example, in the solution to the three variable LP (4.21), the variable x_1 was nonbasic. Suppose the coefficient of x_1 changes from 2 to $2 + \delta$ in the second constraint. Referring to the slack variable coefficients in Table **??**, we note that the current solution remains optimal if $\mathbf{y} = [1, 1]$ satisfies

$$\begin{aligned}\mathbf{y}(A + A_\delta) &= \mathbf{y}\begin{bmatrix} 3 & 1 & 3 \\ 2+\delta & 2 & 3 \end{bmatrix} \\ &= [5 + \delta, 3, 6] \\ &\geq [4, 3, 6],\end{aligned}$$

that is, if $-1 \leq \delta$.

Now we consider the case when the nonzero entry of A_δ corresponds to a basic variable in the LP's solution. For example, suppose in the *FuelPro* LP that an additional δ gallons of stock B is required to produce each gallon of regular unleaded, in which case the coefficient of x_2 in the third constraint of the LP increases from 2 to $2 + \delta$. The tableau corresponding to (4.31) then becomes that shown in Table 4.14.

TABLE 4.14: *Fuelpro* tableau after adjustment to coefficient of A corresponding to a basic variable

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	δ	0	$\frac{1}{2}$	1	46
0	1	δ	0	-1	1	4
0	0	$-\delta$	1	1	-1	4
0	0	$1-\delta$	0	$\frac{3}{2}$	-1	10

To begin the analysis in this situation, we first perform a pivot to update the values of the basic variables and the objective. This pivot leads to the result in Table 4.15.

TABLE 4.15: Updated *Fuelpro* final tableau after adjusting coefficient of x_2 in third constraint

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	0	$\frac{4\delta-1}{2(\delta-1)}$	$\frac{1}{1-\delta}$	$\frac{2(28\delta-23)}{\delta-1}$
0	1	0	0	$\frac{2+\delta}{2(\delta-1)}$	$\frac{1}{1-\delta}$	$\frac{2(7\delta-2)}{\delta-1}$
0	0	0	1	$\frac{2+\delta}{2(1-\delta)}$	$\frac{1}{\delta-1}$	$\frac{2(3\delta+2)}{1-\delta}$
0	0	1	0	$\frac{-3}{2(\delta-1)}$	$\frac{1}{\delta-1}$	$\frac{10}{1-\delta}$

We now make two important observations regarding this tableau. First, in order for the basic variables x_1, x_2 and s_1 to remain nonnegative, it must be the case that $\frac{2(7\delta-2)}{\delta-1}$, $\frac{2(3\delta+2)}{1-\delta}$, and $\frac{10}{1-\delta}$ are all nonnegative. Hence $-\frac{2}{3} \leq \delta \leq \frac{2}{7}$. Second, another iteration of the simplex algorithm is unnecessary provided the nonbasic variable coefficients in the top row of the tableau, $\frac{4\delta-1}{2(\delta-1)}$ and $\frac{1}{1-\delta}$, are nonnegative, i.e., provided $\delta \leq \frac{1}{4}$. Combining these results we conclude that for $-\frac{2}{3} \leq \delta \leq \frac{1}{4}$, the set of basic variables in the solution of the modified is the same as that in the solution of the original. The updated values of the basic variables are given in Table 4.15. Note in particular that when $\delta = 0$, the solution reduces precisely to that of the original *FuelPro* LP.

In summary, when changing an entry in the coefficient matrix A of a standard maximization LP, begin with (4.31).

1. If the entry of A that is changed corresponds to a nonbasic variable in the final solution to the original LP, then duality provides a straightforward

means of determining the range of values for which the current solution remains optimal. No additional pivots are necessary.

2. If the entry of A being changed does correspond to a basic variable, then a pivot is necessary to determine conditions under which all basic variables remain nonnegative and no further simplex algorithm iterations are necessary. This process can be quite tedious.

4.2.4 Performing Sensitivity Analysis with Maple

Sensitivity analysis can be performed with Maple through the use of basic matrix operations. The following worksheet, **Sensitivity Analysis.mw**, illustrates an example, using methods discussed in this section as applied to the *FuelPro* model. In the interest of avoiding redundancy, calculations leading to the original optimal tableau have been omitted. They may be found in the sample worksheet, **Simplex Algorithm.mw**, given at the end of Section 2.1.4. This worksheet begins with the final tableau matrix from **Simplex Algorithm.mw**, denoted by `LP_Matrix`. Recall that **c**, A, **b**, n, and m are all defined at the start of **Simplex Algorithm.mw** and that column 0 and row 0 denote the leftmost column and top row, respectively, of the tableau matrix.

```
> Tableau(LPMatrix);
  # Display final tableau matrix for FuelPro LP.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & 0 & 0 & 0 & \frac{1}{2} & 1 & 46 \\ 0 & 1 & 0 & 0 & -1 & 1 & 4 \\ 0 & 0 & 0 & 1 & 1 & -1 & 4 \\ 0 & 0 & 1 & 0 & \frac{3}{2} & -1 & 10 \end{bmatrix}$$

```
> y:=SubMatrix(LPMatrix,1..1, (m+1)..(m+1+n));
  # Extract from final tableau the slack variable coefficients as a vector.
```

$$y = \begin{bmatrix} 0 & \frac{1}{2} & 1 \end{bmatrix}$$

```
> M:=SubMatrix(LPMatrix,2..(m+1),(m+1)..(m+1+n));
  # Extract from final tableau the submatrix M corresponding to portion
  of tableau below slack variable coefficients.
```

$$M = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 1 & -1 \\ 0 & \frac{3}{2} & -1 \end{bmatrix}$$

```
> ObjectiveCoefficient:=1;
  # Assess sensitivity to first objective coefficient.
```

$$ObjectiveCoefficient = 1$$

```
> e:=UnitVector[row](ObjectiveCoefficient,2);
```

$$e = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

```
> LPMatrixNew:=<UnitVector(1,m+1)|<convert(-c-delta*e,Matrix)+y.A,M.A>|
  <y,M>|<y.b,M.b>>;
  # Create new tableau matrix obtained by perturbing first objective
  coefficient.
```

$$LPMatrixNew = \begin{bmatrix} 1 & -\delta & 0 & 0 & \frac{1}{2} & 1 & 46 \\ 0 & 1 & 0 & 0 & -1 & 1 & 46 \\ 0 & 0 & 0 & 1 & 1 & -1 & 4 \\ 0 & 0 & 1 & 0 & \frac{3}{2} & -1 & 10 \end{bmatrix}$$

```
> Iterate(LPMatrixNew,1,1);
  # Update tableau entries.
```

$$\begin{bmatrix} z & x1 & x2 & s1 & s2 & s3 & RHS \\ 1 & 0 & 0 & 0 & \frac{1}{2}-\delta & 1+\delta & 46+4\delta \\ 0 & 1 & 0 & 0 & -1 & 1 & 4 \\ 0 & 0 & 0 & 1 & 1 & -1 & 4 \\ 0 & 0 & 1 & 0 & \frac{3}{2} & -1 & 10 \end{bmatrix}$$

```
> solve(LPMatrixNew[1,5]>=0 and LPMatrixNew[1,6]>=0, delta);
  # Determine delta values under which nonbasic variable coefficients
  in top row remain nonnegative.
```

$$RealRange = \left(-1, \frac{1}{2}\right)$$

```
> ConstraintNumber:=2;
  # Assess sensitivity to second constraint bound.
```

$$ConstraintNumber = 2$$

```
> e:=UnitVector(ConstraintNumber,m);
```

$$e = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

```
> LPMatrixNew:=<UnitVector(1,m+1)|<convert(-c,Matrix)+y.A,M.A>|
  <y,M>|<y.(b+delta*e),M.(b+delta*e)>>;
  # Create new tableau matrix obtained by perturbing first objective
  coefficient.
```

$$LPMatrixNew = \begin{bmatrix} 1 & 0 & 0 & 0 & \frac{1}{2} & 1 & 46+\frac{1}{2}\delta \\ 0 & 1 & 0 & 0 & -1 & 1 & 46 \\ 0 & 0 & 0 & 1 & 1 & -1 & 4-\delta \\ 0 & 0 & 1 & 0 & \frac{3}{2} & -1 & 10+\frac{3}{2}\delta \end{bmatrix}$$

```
> solve(LPMatrixNew[2,7]>=0 and LPMatrixNew[3,7]>=0 and LPMatrixNew[4,7]>=0,
  delta);
  # Determine conditions under which basic variables remain nonnegative.
```

$$RealRange = (-4, 4)$$

```
> RowNumber:=3:ColumnNumber:=2:
  # Assess sensitivity to entry in
  row 3, column 2 of coefficient matrix.
> Adelta:=Matrix(m,n,{(RowNumber,ColumnNumber)=delta });
  # Create an m by n perturbation matrix.
```

$$A_{delta} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \delta \end{bmatrix}$$

```
> LPMatrixNew:=
  <UnitVector(1,m+1)|<convert(-c,Matrix)+y.(A+Adelta),M.(A+Adelta)>|
  <y,M>|<y.b,M.b>>;
  # Create new tableau matrix that results from perturbing assigned entry
  of A.
```

$$LPMatrixNew = \begin{bmatrix} 1 & 0 & \delta & 0 & \frac{1}{2} & 1 & 46 \\ 0 & 1 & \delta & 0 & -1 & 1 & 4 \\ 0 & 0 & -\delta & 1 & 1 & -1 & 4 \\ 0 & 0 & 1-\delta & 0 & \frac{3}{2} & -1 & 10 \end{bmatrix}$$

```
> Iterate(LPMatrixNew, 3,2);
```

$$\begin{bmatrix} z & x1 & x2 & s1 & s2 & s3 & RHS \\ 1 & 0 & 0 & 0 & \frac{1}{2} - \frac{3}{2}\frac{\delta}{1-\delta} & 1 + \frac{\delta}{1-\delta} & 46 - \frac{10\delta}{1-\delta} \\ 0 & 1 & 0 & 0 & -1 - \frac{3}{2}\frac{\delta}{1-\delta} & 1 + \frac{\delta}{1-\delta} & 4 - \frac{10\delta}{1-\delta} \\ 0 & 0 & 0 & 1 & 1 + \frac{3}{2}\frac{\delta}{1-\delta} & -1 - \frac{\delta}{1-\delta} & 4 + \frac{10\delta}{1-\delta} \\ 0 & 0 & 1 & 0 & 1 + \frac{3}{2}\frac{1}{1-\delta} & -\frac{1}{1-\delta} & \frac{10}{1-\delta} \end{bmatrix}$$

```
> solve( LPMatrixNew[1,5]>=0 and LPMatrixNew[1,6]>=0 and LPMatrixNew[2,7]>=0
  and LPMatrixNew[3,7]>=0 and LPMatrixNew[4,7]>=0,delta);
  # Determine conditions under which no further pivots are necessary
  and basic variables are all nonnegative.
```

$$RealRange\left(\frac{-2}{3}, \frac{1}{4}\right)$$

Exercises Section 4.2

1. Verify formula (4.31).
2. Consider the LP

$$\begin{aligned} &\text{maximize } z = 3x_1 + 2x_2 + x_3 \\ &\text{subject to} \\ x_1 - 2x_2 + 3x_3 &\le 4 \\ x_1 + 3x_2 - 2x_3 &\le 15 \\ 2x_1 + x_2 + x_3 &\le 10 \\ x_1, x_2, x_3 &\ge 0. \end{aligned}$$

(a) Solve this LP. One of the three decision variables in your solution should be nonbasic.

(b) By how much can the objective coefficient of x_1 increase or decrease and the set of basic variables in the solution be unchanged from that of original LP? For this range of coefficient values, what are the values of the decision variables and objective function? Repeat this process, separately, for each of the other objective coefficients.

(c) By how much can the bound in the first constraint increase or decrease and the set of basic variables in the solution be unchanged from that of the original LP? For this range of values, what are the values of all variables, both decision and slack, as well as the objective function? What is the corresponding shadow price? Repeat this process for the bound in the second constraint.

(d) Now consider the LP formed by changing the bounds on the second and third constraints:

$$\begin{aligned} &\text{maximize } z = 3x_1 + 2x_2 + x_3 \\ &\text{subject to} \\ x_1 - 2x_2 + 3x_3 &\leq 4 \\ x_1 + 3x_2 - 2x_3 &\leq 15 + \delta_2 \\ 2x_1 + x_2 + x_3 &\leq 10 + \delta_3 \\ x_1, x_2 &\geq 0. \end{aligned}$$

Sketch the set of ordered pairs, (δ_2, δ_3), for which the set of basic variables in the solution is the same as that of the original LP. For points in this set, what are the values of the decision variables and the objective function in terms of δ_2 and δ_3?

(e) By how much can the coefficient of x_1 in the second constraint increase or decrease and the set of basic variables in the solution be unchanged from that of the original LP? For this range of values, what are the values of all variables, both decision and slack, as well as the objective function?

3. Suppose that *FuelPro* currently produces its two fuels in a manner that optimizes profits but is considering adding a mid-grade blend to its line of products. In terms of the LP, this change corresponds to introducing a new decision variable. Each gallon of mid-grade fuel requires two gallons of stock A and two and a half gallons of stock B and, hence, diverts resources away from producing the other fuel types. The company seeks to determine the smallest net profit per gallon of mid-grade blend that is needed to increase the company's overall profit from its current amount.

(a) The introduction of a third fuel type adds an extra column to the

original coefficient matrix. Determine the new coefficient matrix, $\tilde{A}$.

(b) Suppose that p_m denotes the profit per gallon of the mid-grade fuel type. Then $\tilde{\mathbf{c}} = [4, 3, p_m]$ records the profit per gallon of each of the three fuel types. If $\mathbf{y}_0$ denotes the solution of the original dual LP, calculate $-tilde\mathbf{c} + \mathbf{y}_0\tilde{A}$ in terms of p_m.

(c) Explain why $-tilde\mathbf{c} + \mathbf{y}_0\tilde{A}$ records the decision variable coefficients in the top row of the tableau that result if the same steps of the simplex algorithm used to solve the original LP are applied to the new LP having the third decision variable.

(d) Use the result from the previous question to determine the smallest net profit per gallon of mid-grade blend that is needed to increase the company's overall profit from its current amount.

4.3 The Dual Simplex Method

Theorem 2.4.1 dictates that when we use the simplex algorithm to solve the standard maximization problem, (4.1), the tableau matrix at each iteration takes the general form

$$\begin{bmatrix} 1 & -\mathbf{c} + \mathbf{y}A & \mathbf{y} & \mathbf{y}\mathbf{b} \\ \mathbf{0} & MA & M & M\mathbf{b} \end{bmatrix}. \tag{4.32}$$

At each iteration, the primal LP has a basic feasible solution, whose values are recorded by the entries of $M\mathbf{b}$. Unless the LP is degenerate, all such values are positive. During this process but prior to the termination of the algorithm, at least one entry of $\mathbf{y}$ or $-\mathbf{c} + \mathbf{y}A$ is negative, implying $\mathbf{y}$ is not dual-feasible. At the completion of the algorithm, both $\mathbf{y}A \geq \mathbf{c}$ and $\mathbf{y} \geq \mathbf{0}$, whereby we simultaneously obtain optimal solutions to both the primal LP and its dual.

Unfortunately, if $\mathbf{b}$ has one or more negative entries, then so too can $M\mathbf{b}$. In this case the primal has a solution that is basic, but not basic feasible. The *dual simplex method*, which uses the same tableau as the original method, is well suited for addressing such a situation. At each stage of the algorithm, all entries of $\mathbf{y}$ and $-\mathbf{c} + \mathbf{y}A$ are nonnegative, implying $\mathbf{y}$ is dual-feasible. At the same time, at least one entry of the rightmost tableau column, $M\mathbf{b}$, is negative, so that the basic variable values for the primal at that stage constitute merely a basic, but not basic feasible, solution. The goal of the algorithm then becomes one of making all entries of $M\mathbf{b}$ nonnegative. For when we achieve this outcome, we obtain a basic feasible solution of the primal, whose values are recorded by $M\mathbf{b}$, along with a basic feasible solution, $\mathbf{y}$, of the dual. Since their objective values are both recorded by $\mathbf{y}\mathbf{b}$, both solutions are optimal, and our tableau is identical to that obtained using the regular simplex algorithm.

4.3.1 Overview of the Method

Here is a brief outline of the algorithm.

1. We start with a standard maximization problem (4.1) whose initial tableau matrix is given by

$$\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} & 0 \\ \mathbf{0} & A & I_m & \mathbf{b} \end{bmatrix}. \tag{4.33}$$

2. The process begins by finding an initial basic feasible solution, $\mathbf{y}$, of the dual LP. In other words, we need to find $\mathbf{y} \geq \mathbf{0}$ that satisfies $\mathbf{y}A \geq \mathbf{c}$. In general, finding $\mathbf{y}$ can prove difficult. However, one situation in which

it is always easy to do so occurs when $\mathbf{c} \leq \mathbf{0}$. For in this situation, $\mathbf{y} = \mathbf{0}$ is automatically dual-feasible since $\mathbf{y}A \geq \mathbf{c}$. It is for this reason that the dual simplex is well suited for an LP whose goal is to maximize an objective function whose decision variable coefficients are nonpositive, or, equivalently, to minimize an objective function whose decision variable coefficients are all nonnegative.

3. Unlike the usual simplex algorithm, where we first decide which nonbasic variable is to become basic, i.e., we determined the "entering variable," in the dual simplex algorithm, our first step is to decide which negative basic variable is to become nonbasic, i.e., we determine the "departing basic variable." At the first iteration of the new algorithm, this is accomplished by determining the most negative entry of $\mathbf{b}$.

4. We perform the ratio test in a manner analogous to the usual simplex algorithm in order to determine which nonbasic variable replaces the departing variable in the set of basic variables. The tableau is updated. The process is repeated until all values on the right-hand side of the tableau are nonnegative. That is, the process terminates when we obtain a basic feasible solution to the primal LP.

4.3.2 A Simple Example

To illustrate all details of the algorithm, we consider the following two-variable minimization problem:

$$\begin{aligned} &\text{minimize } w = 3x_1 + 2x_2 \\ &\text{subject to} \\ &2x_1 + 3x_2 \geq 30 \\ &-x_1 + 2x_2 \leq 6 \\ &x1 + 3x_2 \geq 20, \end{aligned} \tag{4.34}$$

whose feasible region is shown in Figure 4.3.

If we change the objective to that of maximizing $z = -w = -3x_1 - 2x_2$, express all constraints in $\leq$ form, and introduce slack variables, s_1, s_2, and s_3, we obtain the tableau shown in Table 4.16.

TABLE 4.16: BV = $\{s_1, s_2, s_3\}$

z	x_1	x_2	s_1	s_2	s_3	RHS
1	3	2	0	0	0	0
0	-2	[-3]	1	0	0	-30
0	-1	2	0	1	0	6
0	-1	-3	0	0	1	-20

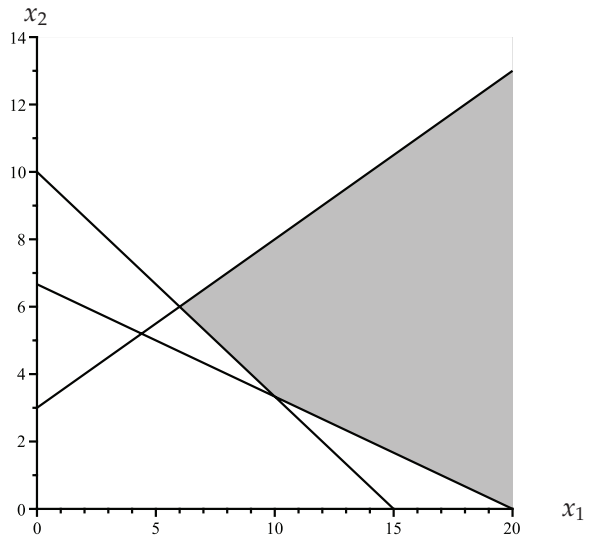


FIGURE 4.3: Feasible region for LP (4.34).

Initially, we choose $s_1 = -30$, $s_2 = 6$, and $s_3 = -20$ as our basic variables. Together, with $x_1 = x_2 = 0$, they constitute a basic, but not basic feasible solution, of (4.34). In terms of Figure 4.3, this initial basic solution corresponds to the origin. In terms of the dual LP, $\mathbf{y} = [0, 0, 0]$, the slack variable coefficients in the top row of Table 4.16, which is dual-feasible.

The tableau shows that $s_1 = -30$ is the most negative basic variable, so it becomes the departing variable. Thus we focus on its corresponding row and must decide which nonbasic variable, x_1 or x_2, replaces s_1. The two equations, $-2x_1 + s_1 = -30$ and $-3x_2 + s_1 = -30$, indicate that we may let s_1 increase to zero by allowing either x_1 to increase to 15 or x_2 to increase to 10. The first of these choices results in $z = -3 \cdot 15 = -45$, the latter in $z = 2 \cdot 10 = -20$. Because our goal is to maximize z, we let x_2 replace s_1 as a basic variable and thus pivot on the highlighted entry in Table 4.16. From this analysis, we conclude that the dual simplex ratio test works as follows: Once we have determined a row corresponding to the departing basic variable, i.e., a pivot row, we determine the column of the entering basic variable as follows. For each negative entry in the pivot row, we compute the ratio of the corresponding entry in the top row (which is necessarily nonnegative by dual-feasibility) divided by the pivot row entry. The smallest of all such ratios, in absolute value, determines the pivot column.

We thus pivot on the highlighted entry in Table 4.16. The resulting tableau is given in Table 4.17. Observe that $x_1 = 0$ and $x_2 = 10$, which corresponds to another basic solution in Figure 4.3.

TABLE 4.17: BV = $\{x_2, s_2, s_3\}$

z	x_1	x_2	s_1	s_2	s_3	RHS
1	$\frac{5}{3}$	0	$\frac{2}{3}$	0	0	-20
0	$\frac{2}{3}$	1	$-\frac{1}{3}$	0	0	10
0	$-\frac{7}{3}$	0	$\frac{2}{3}$	1	0	-14
0	1	0	-1	0	1	10

At this stage, s_2 is the only negative basic variable. By the ratio test applied to the row corresponding to s, we see that x_1 replaces s_2 as a basic variable. The resulting pivot leads to the final tableau in Table 4.18.

TABLE 4.18: BV = $\{x_1, x_2, s_3\}$

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	$\frac{8}{7}$	$\frac{5}{7}$	0	-30
0	0	1	$-\frac{1}{7}$	$\frac{2}{7}$	0	6
0	1	0	$-\frac{2}{7}$	$-\frac{3}{7}$	0	6
0	0	0	$-\frac{5}{7}1$	$\frac{3}{7}$	1	4

The dual simplex algorithm terminates at this stage because all entries in the right-hand side of the tableau, below the top row, are nonnegative. In other words, a basic feasible solution to the primal LP has been obtained. The primal solution is given by $(x_1, x_2) = (6, 6)$ with $z = -30$, which corresponds to $w = 30$ in the original LP (4.34).

If we compare this method with the Big M Method for solving the LP, we see that it provides an elegant and efficient method that completely eliminates the need for introducing artificial variables.

Another setting in which the dual simplex method proves extremely useful is in sensitivity analysis, specifically when one desires to know the effect of adding a constraint to an LP. For example, suppose that *FuelPro* must add a third stock, call it stock C, to each fuel type in order to reduce emissions. Assume that production of each gallon of premium requires 5 gallons of stock C and that production of each gallon of regular unleaded requires 6 gallons of stock C. Furthermore, at most 75 gallons of stock C are available for production. It follows then that we must add the constraint $5x_1 + 6x_2 \leq 75$ to the original *FuelPro* LP. Instead of completely resolving a new LP, we may begin with the final tableau for the original *FuelPro* LP, given by Table 2.4 from Section 2.1, and add a row and column, together with a slack variable s_4 corresponding to the new constraint. The result is shown in Table 4.19.

TABLE 4.19: *FuelPro* tableau after addition of new constraint

z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
1	0	0	0	$\frac{1}{2}$	1	0	46
0	1	0	0	-1	1	0	4
0	0	0	1	1	-1	0	4
0	0	1	0	$\frac{3}{2}$	-1	0	10
0	5	6	0	0	0	1	75

To update the values of the basic variables x_1 and x_2 in the tableau, we must perform two pivots. The resulting tableau is given in Table 4.20.

TABLE 4.20: Updated tableau after pivots are performed

z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
1	0	0	0	$\frac{1}{2}$	1	0	46
0	1	0	0	-1	1	0	4
0	0	0	1	1	-1	0	4
0	0	1	0	$\frac{3}{2}$	-1	0	10
0	0	0	0	-4	1	1	-5

At this stage, $\mathbf{y} = \begin{bmatrix} 0 & \frac{1}{2} & 1 & 0 \end{bmatrix}$ is dual-feasible so that we may apply the dual simplex algorithm directly. Only one iteration is required, in which case we replace a negative basic variable, s_4, with the nonbasic variable s_2. Pivoting on the highlighted entry in Table 4.20 yields the final tableau in Table 4.21.

TABLE 4.21: Updated tableau after one dual-simplex iteration

z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
1	0	0	0	0	$\frac{9}{8}$	$\frac{1}{8}$	$\frac{363}{8}$
0	1	0	0	0	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{21}{4}$
0	0	0	1	0	$-\frac{3}{4}$	$\frac{1}{4}$	$\frac{11}{4}$
0	0	1	0	0	$-\frac{5}{8}$	$\frac{3}{8}$	$\frac{65}{8}$
0	0	0	0	1	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{5}{4}$

From this final tableau, we observe that the addition of the new constraint leads to an updated solution of $(x_1, x_2) = \left(\frac{21}{4}, \frac{65}{8}\right)$ with a corresponding profit of $z = \dfrac{363}{8} = 45.375$. Observe how the added constraint has decreased *FuelPro Petroleum Company*'s profit from the original model.

As the steps involved in applying the dual simplex algorithm are quite similar

to those of its regular counterpart, we should not be surprised that the Maple worksheet, **Simplex Algorithm.mw**, from Section 2.1 is easily modified to execute the dual simplex method. Specifically, we substitute for the `RowRatios` procedure in the original worksheet, a procedure that computes "column ratios" instead. Syntax for doing so is given as follows:

```
> ColumnRatios:=proc(M,r) local k:
  for k from 2 to nops(convert(Row(M,r),list))-1 do
  if M[r+1,k]=0 then print(cat(``Column'', convert(k-1,string),
  ``Ratio Undefined''))
  else print(cat(``Column'',convert(k-1,string), ``Ratio= '',
  convert(evalf(M[1,k]/M[r+1,k]),string))) end if; end do; end:
```

Here we follow our numbering convention that column 0 and row 0 denote the leftmost column and top row, respectively, of the tableau matrix.

Exercises Section 4.3

1. Use the dual simplex algorithm to solve each of the following LPs.

   (a)

$$\begin{aligned} &\text{minimize } z = x_1 + 4x_2 \\ &\text{subject to} \\ &x_1 + 3x_2 \geq 1 \\ &4x_1 + 18x_2 \geq 5 \\ &x_1, x_2 \geq 0 \end{aligned}$$

   (b)

$$\begin{aligned} &\text{minimize } z = x_1 + x_2 \\ &\text{subject to} \\ &x_1 \leq 8 \\ &-x_1 + x_2 \leq 4 \\ &-x_1 + 2x_2 \geq 6 \\ &2x_1 + x_2 \leq 25 \\ &3x_1 + x_2 \geq 18 \\ &-x_1 + 2x_2 \geq 6 \\ &x_1, x_2 \geq 0 \end{aligned}$$

2. Consider again the scenario when a third stock, stock C, is added to each of the fuel types in the *FuelPro* LP. Assume that each gallon of premium requires 5 units of stock C, each unit of regular unleaded requires 6 units of stock C. What is the minimum number of available units of stock C that will change the optimal solution from its value of $(x_1, x_2) = (4, 10)$ in the original LP?

Chapter 5

Integer Linear Programming

5.1 An Introduction to Integer Linear Programming and the Branch and Bound Method

For many linear programming applications, we seek solutions in which all decision variables are integer-valued. We have already discovered that, under certain conditions, the solution of a minimum cost network flow problem has this property. In this section we discover a general technique for solving linear programming problems that require integer-valued solutions. Such an LP is called an *integer linear programming problem*, or ILP.

5.1.1 A Simple Example

To gain a sense of how to solve ILPs, we first consider a simple production problem faced by the *Great Lakes Kayak Company* (GLKC), which manufactures two types of kayaks, one a recreational, i.e., "rec" kayak, suitable for use on inland lakes and calm rivers, and the other a larger sea kayak suitable for open water paddling on the Great Lakes and the ocean. Both kayaks are comprised of two grades of plastic, labeled grades A and B. Each "rec" kayak requires 20 pounds of each grade plastic and sells for $300. The sea kayak, on the other hand, require 10 pounds of grade A and and 30 pounds of grade B and sells for $400. Suppose the daily availability of the plastic is given by 60 pounds for grade A and 90 pounds for grade B. Under the assumption the company sells all the kayaks that it produces, we wish to determine how many kayaks of each type should be produced, subject to the given constraints, so as to maximize daily revenue.

Letting x_1 and x_2 denote the numbers of produced rec kayaks and sea kayaks, respectively, we can easily formulate GLKC's goal as that of solving ILP (5.1):

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\quad \text{subject to} \\ &\qquad 2x_1 + x_2 \leq 6 \\ &\qquad 2x_1 + 3x_2 \leq 9 \\ &\qquad x_1, x_2 \geq 0; x_1, x_2 \in \mathbb{Z}. \end{aligned} \tag{5.1}$$

Here, z represents the revenue, measured in hundreds of dollars. Note, in particular, we require that x_1 and x_2 be integer-valued in the solution.

Figure 5.1 illustrates the candidate solution values of x_1 and x_2 for ILP 5.1, namely the ordered pairs in the shaded region whose entries are integer-valued. We call such candidates, feasible *lattice points*.

One approach to solving ILP 5.1 is of course to merely evaluate the objective function at all feasible lattice points and to determine which produces the largest objective value. If the inequalities that comprise the ILP produce a bounded region, then there exist only a finite number of candidates from which to choose. For small-scale problems, this approach is satisfactory, but for larger-scale problems, it proves extremely inefficient, thereby motivating the need for different methods.

5.1.2 The Relaxation of an ILP

To any ILP, there corresponds an LP, called the *relaxation* of the ILP, which is formed by using the ILP but eliminating the constraint requirement that decision variables be integer-valued. For (5.1), the relaxation solution is straightforward to compute and is given by

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 9/4 \\ 3/2 \end{bmatrix} = \begin{bmatrix} 2.25 \\ 1.5 \end{bmatrix} \text{ and } \quad z = \frac{51}{4} = 12.75. \tag{5.2}$$

Because (5.1) is a maximization problem and because the constraints in (5.1) are more restrictive than those in the relaxation, we may be assured that the objective value in the solution of (5.1) is no larger than $z = 12.75$. This principle holds in general: The optimal objective value of an ILP involving maximization is no larger than that of the corresponding relaxation LP. Consequently, if the relaxation has a solution whose decision variables happen to be integer-valued, then this solution coincides with that of the original ILP. As we shall discover, this fact proves crucial for developing a means of solving ILPs.

Usually, the relaxation produces a solution in which decision variables are not integer-valued. Figure 5.1 illustrates such an outcome for the GLKC LP.

While the shaded portion represents the feasible region for the relaxation LP, only the lattice points in this feasible region are also feasible for the ILP.

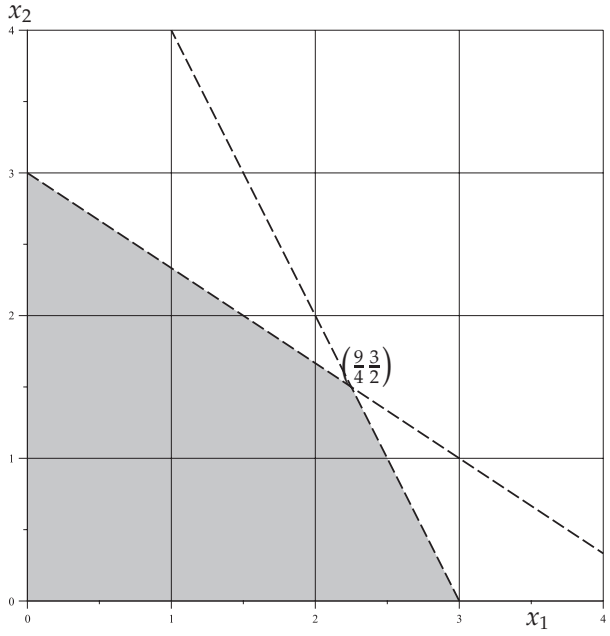


FIGURE 5.1: Feasible region and solution for the GLKC ILP relaxation.

A naive approach to solving the ILP consists of rounding off the relaxation solution to the nearest lattice point. Unfortunately, this approach can lead to decision variable values that do not correspond to the optimal solution or are infeasible to begin with.

Waypoint 5.1.1. Determine the solution of ILP (5.1) by graphical inspection of Figure 5.1. How does this solution compare to that of the relaxation or to that obtained by rounding off the relaxation solution?

5.1.3 The Branch and Bound Method

The *branch and bound method* is a technique for solving an ILP, which is based upon solving a carefully chosen sequence of closely related LP's. The first step of the method requires solving the corresponding relaxation LP. If the solution (x_1, x_2) of the relaxation is integer valued, then this solution coincides with that of the ILP. If not, the original feasible region is partitioned into two

disjoint (i.e., non-overlapping) regions, neither of which contains (x_1, x_2). Each of these two regions gives rise to a new LP, which is merely the original LP together with an additional constraint. These two new LPs are then solved, a process we shall refer to as *branching*. If an integer-valued solution arises from one of these new LPs, then it becomes a candidate solution for the original ILP. If an integer-valued solution does not arise, then the corresponding region is again partitioned, and the process repeats itself. Eventually, all cases are exhausted, and we obtain an optimal solution for the ILP.

We illustrate the technique by applying it to the GLKC ILP. The solution of the relaxation is given by

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.25 \\ 1.5 \end{bmatrix} \text{ and } \quad z = 12.75.$$

Neither of the decision variables in this solution is integer-valued. To partition the feasible region into two disjoint regions neither of which contains $(2.25, 1.5)$, we have a choice of *branching* on either of the decision variables in a way that removes the given solution value from consideration in a newly designed LP.

By *branching* at a particular stage on a decision variable x_i in an LP's solution, we simply mean we solve two new LPs. If $\tilde{x}_i$ denotes the value of the decision variable x_i in the LPs solution, which is not integer-valued at that stage, the first new LP is the LP at that stage, together with the added constraint that x_i is greater than or equal to the smallest integer greater than $\tilde{x}_i$. The second LP is defined analogously, but the added constraint is instead that x_i is less than or equal to the largest integer less than $\tilde{x}_i$.

Here we elect to branch on x_1. To eliminate $x_1 = 2.25$ from arising in a new solution, we consider two new LPs:

$$\begin{aligned} \text{maximize } z &= 3x_1 + 4x_2 \\ \text{subject to} & \\ \text{Original constraints plus} & \\ x_1 &\leq 2 \\ x_1, x_2 &\geq 0 \end{aligned} \tag{5.3}$$

and

$$\begin{aligned} \text{maximize } z &= 3x_1 + 4x_2 \\ \text{subject to} & \\ \text{Original constraints plus} & \\ x_1 &\geq 3 \\ x_1, x_2 &\geq 0. \end{aligned} \tag{5.4}$$

Note that the first of these LPs is nothing more than our original LP using that portion of its feasible region in Figure 5.2 that falls to the left of $x_1 = 2$. The second LP uses that portion to the right of $x_1 = 3$.

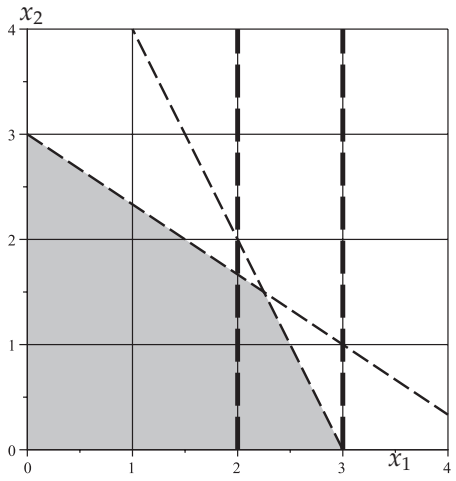


FIGURE 5.2: Branching on x_1 in the feasible region of the relaxation LP.

Solving LPs (5.3) and (5.4) we obtain the following solutions, respectively:

1. $x_1 = 2, \quad x_2 = 1.\bar{6}, \quad$ and $\quad z = 12.\bar{6}$;

2. $x_1 = 3 \quad x_2 = 0, \quad$ and $\quad z = 9$.

We now branch again by using these results and adding more constraints to the corresponding LPs. However, the solution of (5.4) is integer-valued, so adding more constraints to (5.4) can only produce an LP whose optimal objective value is no greater than 9. For this reason, we label $\mathbf{x} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$, which corresponds to $z = 9$, a *candidate* for the optimal solution of (5.2), and we branch no further on LP (5.4).

The solution of (5.3) produces a value of x_2 that is not integer-valued, so we branch on x_2 and form two new LPs that arise by eliminating $x_2 = 1.\bar{6}$ from consideration. These are given as follows:

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\text{subject to} \\ &\text{Original constraints plus} \\ &x_1 \le 2 \\ &x_2 \le 1 \\ &x_1, x_2 \ge 0 \end{aligned} \tag{5.5}$$

and

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\text{subject to} \\ &\text{Original constraints plus} \\ &x_1 \le 2 \\ &x_2 \ge 2 \\ &x_1, x_2 \ge 0. \end{aligned} \tag{5.6}$$

The respective solutions of (5.5) and (5.6) are as follows:

1. $x_1 = 2, \quad x_2 = 1, \quad \text{and} \quad z = 10;$
2. $x_1 = 1.5, \quad x_2 = 2, \quad \text{and} \quad z = 12.5.$

These results indicate that $\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is also a candidate solution. Because its objective value is larger than that produced by the first candidate solution, $\mathbf{x} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$, we eliminate the first candidate from further consideration.

The solution of (5.6) produces a value of x_1 that is not integer-valued, so we branch on x_1, creating two new LPs that arise by eliminating $x_1 = 1.5$ from consideration:

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\text{subject to} \\ &\text{Original constraints plus} \\ &x_1 \le 2 \\ &x_2 \ge 2 \\ &x_1 \le 1 \\ &x_1, x_2 \ge 0 \end{aligned} \tag{5.7}$$

and

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\text{subject to} \\ &\text{Original constraints plus} \\ &x_1 \leq 2 \\ &x_2 \geq 2 \\ &x_1 \geq 2 \\ &x_1, x_2 \geq 0. \end{aligned} \tag{5.8}$$

Clearly, the first constraint in (5.7) is redundant in light of the newly added, more restrictive third constraint. Moreover, the first and third constraints in (5.8) can be combined into the equality constraint, $x_1 = 2$. We only express the constraints for each LP in this manner so as to emphasize the manner in which we are adding more and more constraints to the original LP at each stage of branching.

The second of the preceding LPs is infeasible. The first has its solution given by

$$x_1 = 1, \quad x_2 = 2.\bar{3}, \quad \text{and} \quad z = 12.\bar{3}.$$

Branching on x_2 in (5.7) gives two more LPs:

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\text{subject to} \\ &\text{Original constraints plus} \\ &x_1 \leq 2 \\ &x_2 \geq 2 \\ &x_1 \leq 1 \\ &x_2 \leq 2 \\ &x_1, x_2 \geq 0 \end{aligned} \tag{5.9}$$

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \qquad (5.10)\\ &\text{subject to}\\ &\text{Original constraints plus}\\ &x_1 \le 2\\ &x_2 \ge 2\\ &x_1 \le 1\\ &x_2 \ge 3\\ &x_1, x_2 \ge 0. \end{aligned}$$

LPs (5.9) and (5.10) produce solutions of $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$, respectively, with corresponding objective values of $z = 11$ and $z = 12$. Both are candidate solutions, so we branch no further.

At this stage the branching process is complete, and we have determined four candidate solutions exist. The first of these, $\mathbf{x} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$, has already been eliminated from consideration. Of the remaining three, $\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, and $\mathbf{x} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$, the last yields the largest objective value of $z = 12$. Thus, ILP (5.1) has its solution given by $\mathbf{x} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ so that GLKC should produce no "rec" kayaks and three sea kayaks.

The branching process outlined in the preceding discussion can be visualized through construction of a *tree diagram* as shown in Figure 5.3. When passing from one LP to an LP in a lower-level branch, we merely add a new constraint.

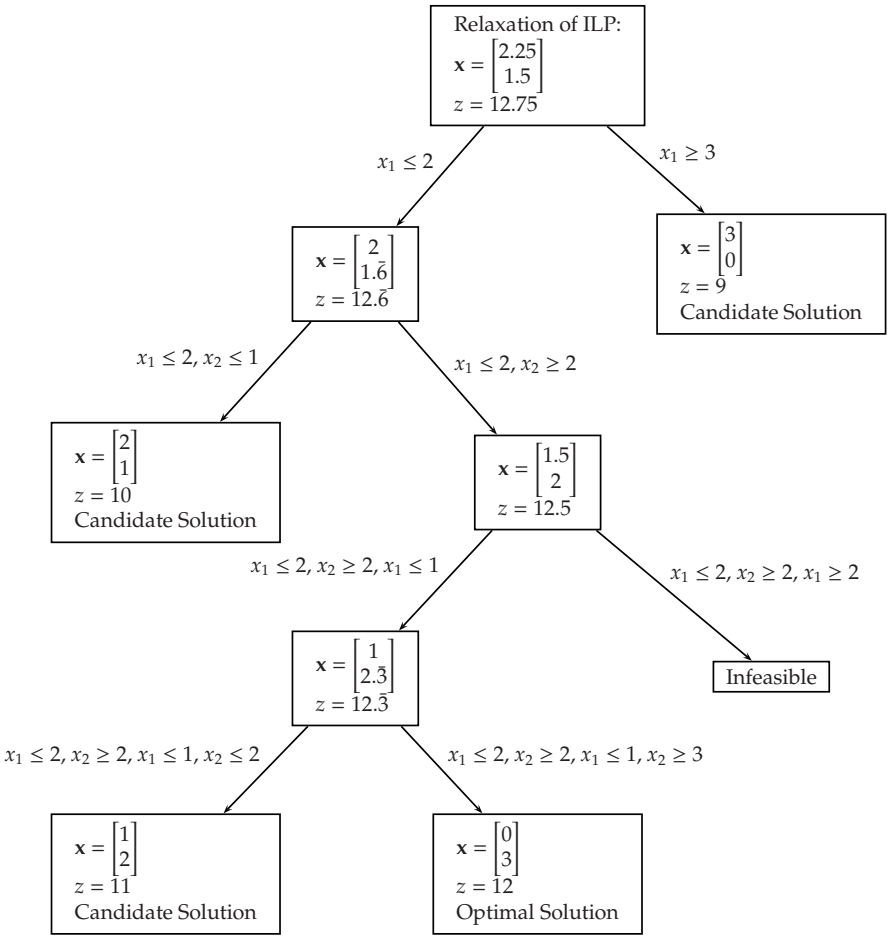


FIGURE 5.3: Tree diagram for ILP (5.1).

Here are some points to bear in mind when using the branch and bound method.

1. Create a tree diagram with nodes and arrows. Each node should clearly convey the LP being solved, together with its optimal solution, especially the objective function value.
2. Along each branch, or edge, of the tree diagram connecting two nodes, indicate the constraint that is added to the LP at the top node in order to formulate the LP of the bottom node.
3. Before branching on a particular variable, pause to consider whether doing so is absolutely necessary. Remember that in a maximization (resp. minimization) problem, objective values can only decrease (resp. increase) from one node to a lower node connected along a particular branch.
4. At a certain point in the solution process, all possible cases along a branch become exhausted and other previously constructed branches must be examined for candidate solutions. *Backtracking*, the process of revisiting unresolved branches in exactly the reverse order they were created, is a systematic means for carrying out this process.

5.1.4 Practicing the Branch and Bound Method with Maple

Practicing the branch and bound method is straightforward using Maple's `LPSolve` command. We first solve the relaxation LP and then progressively add more inequalities to the constraint list until the ILP is solved. Here is syntax from a worksheet, **Practicing the Branch and Bound Method.mw** that solves the relaxation of the GLKC ILP, along with the LP formed by adding the inequality $x_1 \geq 3$ to the relaxation at the first branch.

```
> restart:with(LinearAlgebra):with(Optimization):
> c:= Vector[row]([3,4]);
  # Enter objective coefficients.
```

$$c := \begin{bmatrix} 3 & 4 \end{bmatrix}$$

```
> A:=Matrix(2,2,[2,1,2,3]);
  # Enter constraint matrix.
```

$$A := \begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix}$$

```
> b:=<6,9>;
  # Enter constraint bounds.
```

$$b := \begin{bmatrix} 6 \\ 9 \end{bmatrix}$$

```
> x:=<x1,x2>;
  # Create vector of decision variables.
```

$$x := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

```
> ConstraintMatrix:=A.x-b:
> Constraints:=seq(ConstraintMatrix[i]<=0,i=1..2);

  # Form constraints by extracting components of Ax-b as a sequence
  and setting each component less than or equal to 0.
```

$$constraints := 2x_1 + x_2 - 6 <= 0, 2x_1 + 3x_2 - 9 <= 0$$

```
> LPSolve(c.x,[constraints],assume='nonnegative','maximize'); #
  Solve relaxation.
```

$$[12.750000000000, [x_1 = 2.25000000000000000, x_2 = 1.50000000000000022]]$$

```
> LPSolve(c.x,[constraints,x1 >=3],assume='nonnegative','maximize');
  # Add an inequality to form a new list of constraints. Then solve
  the resulting LP.
```

$$[9, [x1 = 3, x2 = 0]]$$

Remaining LPs that result from further branching are handled in a similar manner so as to obtain the final solution of $\mathbf{x} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$.

5.1.5 Binary and Mixed Integer Linear Programming

Two very important subclasses of ILPs arise frequently in real-world applications. The first consists of *binary linear programming problem*, or BLPs, in which each decision variable must assume the value of zero or one in the final solution. The second is comprised of *mixed integer linear programming problems*, or MILPs, in which some, but not all decision variables must be integer-valued in the solution. Solving mixed MILPs is easily accomplished using the branch and bound method, with branching done only on those variables required to be integer-valued.

For example, consider the mixed MILP,

$$\begin{aligned} &\text{maximize } z = x_1 + 3x_2 + 3x_3 \\ &\quad \text{subject to} \\ &x_1 + 3x_2 + 2x_3 \leq 7 \\ &2x_1 + 2x_2 + x_3 \leq 11 \\ &x_1, x_2, x_3 \geq 0; x_1, x_3 \in \mathbb{Z}. \end{aligned} \tag{5.11}$$

The solution to the relaxation of (5.11) is given by $\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 3.5 \end{bmatrix}$ with corresponding objective value, $z = 31.5$. Since only x_1 and x_3 must be integer-valued, we restrict all branching to these two variables. As the former is already integer-valued in the relaxed solution, we branch on x_3 and consider the original constraints together with two different possibilities for x_3: $x_3 \leq 3$ and $x_3 \geq 4$. The former case leads to a candidate solution for the MILP; the latter leads to infeasibility. The candidate solution is optimal and is labeled in Figure 5.4.

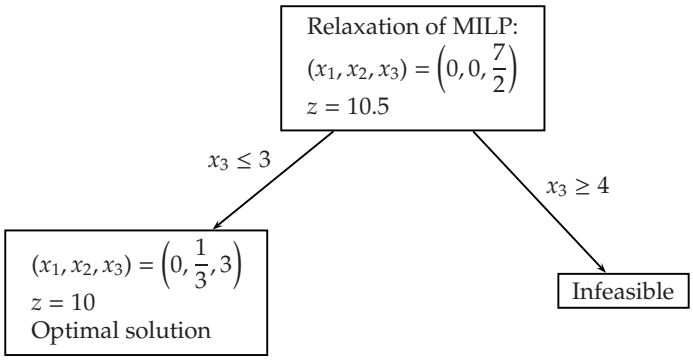


FIGURE 5.4: Tree diagram for MILP (5.11).

5.1.6 Solving ILPs Directly with Maple

While Maple's `LPSolve` command is useful for practicing the branch and bound method, the command already has built-in options for solving ILPs directly. To solve an ILP having no sign restrictions, simply add the option *`assume=integer`*. For an ILP requiring nonnegative decision variable values, add *`assume=nonnegint`* instead. For a MILP, the option `integervariables=[`*`list`*`]` specifies which particular variables are required to take on integer variables. For a BLP, adding *`assume=binary`* requires all decision variables equal to 0 or 1 in the solution, whereas `binaryvariables=[`*`list`*`]` specifies which subset of the decision variables have binary values. Finally, the option *`depthlimit`*`=n` specifies the number of branching levels used by Maple in its solution process. For the GLKC ILP, four branching levels were required. Since Maple's default number is three, an error message is returned unless the *`depthlimit`* is assigned a sufficiently large value.

Here is a sample worksheet illustrating the use of these command options. In the first example, Maple solves the GLKC ILP (5.1) using the matrix inequality form as discussed in Section 1.1.3. In the second, Maple solves MILP (5.11).

```
> restart:with(LinearAlgebra):with(Optimization):
> c:=Vector[row]([3,4]);
  # Create vector of objective coefficients.
```

$$c := \begin{bmatrix} 3 & 4 \end{bmatrix}$$

```
> A:=Matrix(2,2,[2,1,2,3]);
  # Matrix of constraint coefficients.
```

$$A := \begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix}$$

```
> b:=<6,9>;
  # Constraint bounds.
```

$$b := \begin{bmatrix} 6 \\ 9 \end{bmatrix}$$

```
> LPSolve(c, [A, b], assume = nonnegint, 'maximize', depthlimit
  = 5);
```

$$\begin{bmatrix} 12 & \begin{bmatrix} 0 \\ 3 \end{bmatrix} \end{bmatrix}$$

```
> LPSolve(x1+3x2+3x3,[x1+3x2+2x3<=7, 2x1+2x2+x3<=11],assume=nonnegative,
  integervariables=[x1,x3],'maximize');
  # Solve the LP, specifying x1 and x3 are integer-valued.
```

$$[10.000000000000, [x_1 = 0, x_2 = .333333333333333315, x3 = 3]]$$

5.1.7 An Application of Integer Linear Programming: The Traveling Salesperson Problem

Integer linear programming has an enormous, diverse number of applications. One of the most widely studied problems in this area consists of the "Traveling Salesperson Problem," or TSP.

The TSP seeks to determine the minimum distance an individual must travel in order to begin at one location, pass through each of a list of other intermediate locations exactly one time, and return to the starting point. Such a round trip is known as a *tour*.

We will consider the general setting consisting of n destinations, where $n \geq 2$. Figure 5.5 illustrates an example of a tour for the particular case of $n = 4$. We can represent this example through the notation $1 \to 3 \to 2 \to 4 \to 1$.

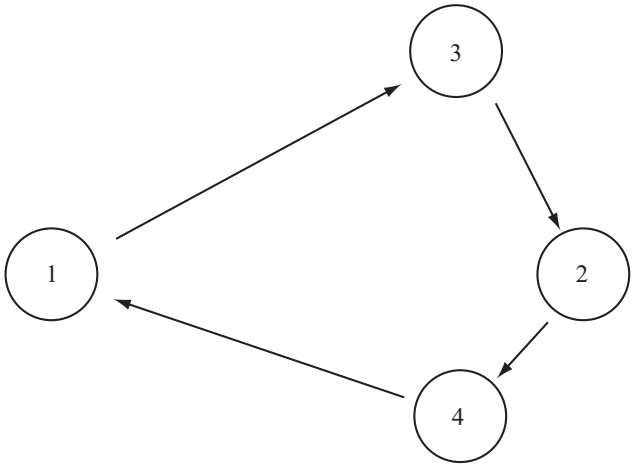


FIGURE 5.5: Tour consisting of 4 destinations.

To solve the TSP, we first define the binary decision variables, x_{ij}, where $1 \leq i, j \leq n$. A value of $x_{ij} = 1$ in the final solution indicates that travel takes place from destination i to j; a value of 0 indicates that no such travel takes place. That no travel takes place from a destination to itself dictates that the problem formulation must guarantee a solution in which $x_{ii} = 0$, for $1 \leq i \leq n$.

Each location is the starting point from which one travels to a new destination, a condition expressed as $\sum_{j=1}^{n} x_{ij} = 1$ for $1 \leq i \leq n$. Similarly, each destination is also ending point of travel from a previous destination so that $\sum_{j=1}^{n} x_{ji} = 1$ for $1 \leq i \leq n$. We will denote the distance between locations i and j, where $1 \leq i, j \leq n$, by d_{ij}. Of course, $d_{ij} = d_{ji}$, and to ensure that $x_{ii} = 0$ for $1 \leq i \leq n$, we set each $d_{ii} = M$, where M is a number much larger than any distance between two different destinations. With this notation, the total distance traveled is given by $D = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$.

At first glance, it might then appear that the solution to the TSP is the same as that obtained by solving the BLP

$$\text{minimize } D = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{5.12}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1, \quad 1 \leq i \leq n$$

$$\sum_{j=1}^{n} x_{ji} = 1, \quad 1 \leq i \leq n$$

$$x_{i,j} \in \{0, 1\}, \quad 1 \leq i, j \leq n.$$

Unfortunately, (5.12) possesses a major problem. Namely, it includes as feasible solutions, decision variable values that do not correspond to tours, but instead to combinations of *subtours*. A *subtour* is a tour of a proper subset of the set of destinations.

Figure 5.6 illustrates an example consisting of two subtours of $n = 4$ destinations. In terms of notation, $1 \rightarrow 3 \rightarrow 1$ and $2 \rightarrow 4 \rightarrow 2$. The decision variable values equal to 1 are precisely x_{13}, x_{31}, x_{24}, and x_{42}. These values satisfy the feasibility criteria in (5.12), but they do not form a tour.

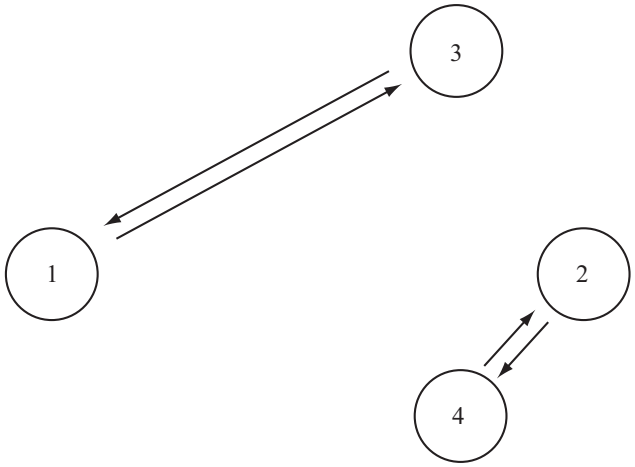


FIGURE 5.6: Two subtours of 4 destinations.

To guarantee that only tours, and not subtours, are feasible, we must introduce more decision variables and use these to add additional constraints to (5.12). We label these decision variables as p_i, where $1 \leq i \leq n$, and let each p_i denote

the position of destination i in a tour. For example, if $1 \to 3 \to 2 \to 4 \to 1$, then $p_1 = 1$, $p_3 = 2$, $p_2 = 3$, and $p_4 = 4$. Note that for each i, $1 \le p_i \le n$. Furthermore, to solve the TSP, we may assume we start at location 1, meaning $p_1 = 1$.

As we shall discover, a family of additional constraints that guarantees only tours, and not combinations of subtours, are feasible in (5.12) can be formulated as

$$p_i - p_j + 1 \le (n-1)(1 - x_{ij}) \text{ where } 2 \le i, j \le n. \tag{5.13}$$

To verify that all tours satisfy (5.13), we consider two cases for each pair of distinct destinations, i and j, in a tour, where $2 \le i, j \le n$. In the first case $x_{ij} = 1$, meaning direct travel takes place from destination i to j and $p_j - p_i = 1$. Therefore,

$$\begin{aligned} p_i - p_j + 1 &= 0 \\ &= (n-1)(1 - x_{ij}) \end{aligned}$$

so that (5.13) holds. In the second case $x_{ij} = 0$, and no such direct travel takes place. Since $p_1 = 1$ and $2 \le i, j \le n$, we have $p_i - p_j \le n - 2$ so that $p_i - p_j + 1 \le n - 1$. Thus, (5.13) holds for the second case as well.

We now verify that values of decision variables corresponding to a combination of subtours passing through all destinations do not satisfy (5.13). To do so, we assume to the contrary that a combination of subtours exists, satisfying (5.13), and, from there, we establish a contradiction.

We begin by selecting the subtour not containing the first destination. Then, for some k satisfying $2 \le k \le n-1$ and for some subset $\{i_1, i_2, \ldots, i_k\}$ of distinct elements of $\{2, \ldots, n\}$,

$$i_1 \to i_2 \to i_3 \to \ldots \to i_k = i_1.$$

By repeated application of (5.13), we obtain the $k-1$ inequalities

$$\begin{aligned} p_{i_1} - p_{i_2} + 1 &\le (n-1)(1 - x_{i_1 i_2}), \\ p_{i_2} - p_{i_3} + 1 &\le (n-1)(1 - x_{i_2 i_3}), \\ \vdots \quad &\le \quad \vdots \\ p_{i_{k-1}} - p_{i_k} + 1 &\le (n-1)(1 - x_{i_{k-1} i_k}). \end{aligned} \tag{5.14}$$

Because $x_{i_m i_{m+1}} = 1$ for $m = 1, 2, \ldots, k-1$, the sum of the $k-1$ inequalities in (5.14) simplifies to

$$p_{i_1} - p_{i_k} + (k-1) \le 0. \tag{5.15}$$

However, $i_1 = i_k$ so that $p_{i_1} = p_{i_k}$, from which it follows $k \le 1$. But this result contradicts our assumption $k \ge 2$. We may therefore conclude that decision variables corresponding to a subtour do not satisfy constraints (5.13).

Combining all these results, we finally obtain an ILP formulation of the TSP:

$$\begin{aligned}
&\text{minimize } D = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} && (5.16)\\
&\text{subject to}\\
&\sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le i \le n\\
&\sum_{j=1}^{n} x_{ji} = 1, \quad 1 \le i \le n\\
&p_i - p_j + 1 \le (n-1)(1-x_{ij}), \quad 2 \le i, j \le n\\
&p_1 = 1\\
&2 \le p_i \le n, \quad 2 \le i \le n\\
&p_i \in \mathbb{Z}, \quad 1 \le i \le n.\\
&x_{i,j} \in \{0,1\}, \quad 1 \le i, j \le n.
\end{aligned}$$

Waypoint 5.1.2. Jane wishes to ride her bicycle on a tour passing through five towns. The names of the towns and the distances between them are summarized in Table 5.1. Assume that Jane's ride starts in Jamestown. Use Maple to determine the tour that minimizes her total distance traveled. Hint: Define the two families of variables in (5.16) using arrays:

```
> x:=array(1..5,1..5):
> p:=array(1..5):
```

The first two families of constraints can be formulated by combining the `add` and `seq` commands. For the third family of constraints, next two `seq` commands as follows:

```
> seq(seq(p[i]-p[j]+1<=4*(1-x[i,j]),j=2..n),i=2..n):
```

Your results should indicate that Jane rides from Jamestown to Cutlerville to Caledonia to Dorr to Byron Center, and back to Jamestown.

While (5.16) provides a means of stating the TSP as an ILP, the task of solving it proves inefficient, even for problems involving a moderate number of

TABLE 5.1: Distances between towns for Jane's bicycle ride

City	Jamestown	Dorr	Byron Center	Caledonia	Cutlerville
Jamestown	■	13	7	22	11
Dorr	13	■	6	15	11
Byron Center	7	6	■	13	6
Caledonia	22	15	13	■	12
Cutlerville	11	11	6	12	■

decision variables. For this reason, determining more efficient methods for solving larger-scale TSPs remains a much-pursued goal in the field of linear programming.

Exercises Section 5.1

1. Use the branch and bound method to solve each of the following ILPs.

(a)

$$\begin{aligned} &\text{maximize } z = 2x_1 + 3x_2 \\ &\quad \text{subject to} \\ &\quad 2x_1 + x_2 \leq 7 \\ &\quad x_1 + 2x_2 \leq 7 \\ &\quad x_1, x_2 \geq 0; x_1, x_2 \in \mathbb{Z} \end{aligned}$$

(b)

$$\begin{aligned} &\text{maximize } z = 5x_1 + 6x_2 \\ &\quad \text{subject to} \\ &\quad 2x_1 + 3x_2 \leq 18 \\ &\quad 2x_1 + x_2 \leq 12 \\ &\quad 3x_1 + 3x_2 \leq 20 \\ &\quad x_1, x_2 \geq 0; x_1, x_2 \in \mathbb{Z} \end{aligned}$$

(c)

$$\begin{aligned} \text{maximize } z &= 4x_1 + 3x_2 + 5x_3 \\ \text{subject to} & \\ x_1 + x_2 + x_3 &\leq 17 \\ 6x_1 + 3x_2 + 3x_3 &\leq 22 \\ 3x_1 + 3x_2 \quad &\leq 13 \\ x_1, x_2, x_3 &\geq 0; x_1, x_2, x_3 \in \mathbb{Z} \end{aligned}$$

2. Solve the mixed-integer linear programming problem that arises by restricting only x_2 and x_3 to be integers in 1c.

3. To what digits (0-9) can one assign each of the letters in the phrase below to make the equation mathematically valid?

$$\text{one} + \text{one} + \text{two} + \text{two} + \text{three} + \text{eleven} = \text{twenty} \tag{5.17}$$

(Hint: Note that only ten letters of the alphabet are used in this statement so that we may number the letters. For example, suppose we number "o", "n", and "e" as 1, 2, and 3, respectively. If we define x_{ij}, where $1 \leq i \leq 10$ and $0 \leq j \leq 9$ is a binary decision variable whose solution value is 1 provided letter i corresponds to digit j, then the letter "o" is represented by the sum, $\sum_{j=0}^{9} jx_{1j}$, the letter "n" by the sum $\sum_{j=0}^{9} jx_{2j}$, and the letter "e" by the sum $\sum_{j=0}^{9} jx_{3j}$. This means that the word "one" is represented by the sum

$$100\sum_{j=0}^{9} jx_{1j} + 10\sum_{j=0}^{9} jx_{2j} + \sum_{j=0}^{9} jx_{3j}.$$

Other words are constructed in a similar manner. Constraints arise from Equation (5.17), together with the fact that each digit corresponds to exactly one letter and that equation. Since the goal is merely one of determining feasible values for these variables, the objective function can be chosen as a constant and the goal as either maximization or minimization.)

4. Suppose Jane wishes to bring snacks high in carbohydrate content with her as she completes her bicycle ride. Weights and carbohydrate content for her five choices are summarized in Table 5.2. If Jane believes she can only afford to bring 100 grams of food with her as she rides, how many of each snack should she bring so as to maximize her total carbohydrate

intake? What if Jane can bring 125 grams instead? 150 grams? (Note: An IP having only one constraint, such as this, is an example of a *knapsack problem*.)

TABLE 5.2: Weight and nutritional data taken from manufacturer's web sites

Item	Cereal Bar	Powerbar ™	Cliff Bar ™	Luna Bar ™	GU Gel ™
Carbs (gms.)	29	43	42	27	25
Weight (gms.)	42	65	68	48	32

5. Coach Anderson must decide his baseball team's pitching rotation.[1] The rotation will consist of four starting pitchers, one mid-innings relief itcher, and one "closer," (i.e., late inning pitcher). Coach Sparky has six pitchers, whose earned-run-averages (ERAs) in these various roles are summarized in Table 5.3.

TABLE 5.3: Pitching data for Coach Anderson's team

Pitcher/E.R.A.	Starting	Mid-Innings Relief	Closer
Clay	3.5	2.8	2.7
Don	2.42	2.45	2.8
Gary	3.16	3.2	3.6
Jack	2.5	2.6	2.8
Pedro	2.7	2.8	2.95
Rawley	3	2.7	2. 6

If each pitcher serves in exactly one role and if the average combined earned-run-average is the average of the six pitchers' earned-run-averages in their assigned roles, what pitching rotation should Coach Anderson choose in order to minimize the average combined earned-run-average?

6. The *n-Queens Problem* asks for the largest number of queens that can be placed on an n-by-n chessboard so that no two queens are capable of attacking one another.[2] Construct a BLP that solves the 8-Queens Problem. That is, solve the problem for the standard size chessboard. (Hint: Let x_{ij}, where $1 \leq i, j \leq 8$, denote a binary decision variable whose solution value equals 1 provided a queen occupies the entry in row i, column j of the chessboard. Then the sum of the decision variables along each column, each row, and each diagonal cannot exceed 1.)

7. Suppose m is a positive integer, and let $n = m^2$. In the n-by-n *Sudoku*

[1] Based upon Machol, [25], (1970).
[2] Based upon Letavec and Ruggiero, [22], (2002).

puzzle, a player is given an n-by-n grid comprised of n, m-by-m non-overlapping sub-grids.[3] A digit between 1 and n appears in certain grid entries. The goal of the puzzle is to enter a digit between 1 and n in each of the remaining grid entries so that each row, each column, and each m-by-m sub-grid contains exactly one of the n digits. An example of a 4-by-4 puzzle is depicted in Table 5.4. In this case $m = 2$ and $n = 4$.

TABLE 5.4: Example of a 4-by-4 Sudoku puzzle

	1		
		2	
			4
3			

(a) Formulate a BLP that solves the n-by-n Sudoku puzzle. To get started, let x_{ijk}, where $1 \leq i, j, k \leq n$, denote a binary decision variable, whose solution value equals 1 provided the entry in row i, column j of the puzzle equals k, and zero otherwise. Let the objective of the BLP be any constant function; the constraints correspond to feasible puzzle solutions and can be broken down as follows:

i. Each column contains one entry equal to k, where $1 \leq k \leq n$.

ii. Each row contains one entry equal to k, where $1 \leq k \leq n$.

iii. Every position in the puzzle must be filled.

iv. Certain puzzle entries are known from the outset. This can be stated as a sequence of values, $x_{ijk} = 1$, where each (i, j, k) is a triple satisfying $1 \leq i, j, k \leq n$.

v. Each m-by-m sub-grid contains exactly one of the n digits. (Hint: Let $B_{p,q}$, where $1 \leq p, q \leq m$, denote a sub-grid of the puzzle. Then the entries of this sub-grid can be indexed by ordered pairs, (i, j), where $mp - m + 1 \leq i \leq mp$ and $mq - m + 1 \leq j \leq mq$.)

(b) Use your result to solve the puzzle shown in Table 5.4.

8. When an individual writes numerous checks, the possibility exists that his or her bank will present several checks for payment against his or her account on the same day.[4] Depending upon the individual's account balance and the sequence in which these checks are processed, varying amounts of insufficient fund (NSF) fees can be collected by the bank. Some banks have used sequencing policies that maximize their total collected NSF fees, a policy that has resulted in several lawsuits.

[3] Based upon Bartlett, et al., [3], (2008).

[4] Based upon Apte, et al.,[1], (2004).

Consider an example in which Joe Freshman has a balance of $1200 in his banking account, which charges a $20 NSF fee per bounced check. On a particular day, the bank receives seven checks that it presents against Joe Freshman's account. These checks are for the following amounts: $30, $80 ,$140, $160, $500, $600, and $800.

(a) Under a low-high policy, checks presented on the same day are processed in the order of lower to higher check amounts. Calculate the total amount of NSF fees the bank charges Joe under this policy.

(b) Under a high-low policy, checks presented on the same day are first listed in order of descending amounts. Starting at the top of the list, the bank ascertains whether the current balance exceeds the given check amount. If so, the bank clears the check by deducting its amount from the current balance and then moves to the next check on this list. If not, the bank charges an NSF fee before proceeding to the next check, which is of smaller amount and therefore may not necessarily bounce. Calculate the total amount of NSF fees the bank charges Joe under this policy. Your value should be larger than that charged in (a).

To determine which strategy is optimal for Joe and which is optimal for the bank, we let c_i represent the amount of check i and let x_i, where $1 \leq i \leq 7$, denote a binary decision variable whose solution value equals 0 provided check i "bounces" and equals 1 if the check "clears."

(c) Formulate an expression for the total NSF fees collected by the bank.

(d) Determine a constraint that represents the fact that the account balance is at least as large as the sum of the cleared checks.

(e) Set up and solve a BLP that determines the order in which checks should be processed so as to minimize the total NSF fees. How does this result compare with the low-high policy?

(f) Now construct a new BLP that spells out the order in which the bank should process the checks so as to maximize the NSF fees it collects. Your objective function will be the same as that in the previous question. However, your goal will now be to maximize as opposed to minimize. To the previous BLP you will also need to add seven new constraints, which help the bank identify checks that can be "made to bounce," so to speak. By considering the separate cases, $x_j = 0$ and $x_j = 1$, explain why the following constraints assist the bank in achieving this goal:

$$1200x_j + c_j \geq (1200 + .01) - \sum_{i=1}^{7} c_i x_i, \text{ where } 1 \leq j \leq 7.$$

(g) How does the solution of the BLP from the previous question compare with that of the high-low policy?

5.2 The Cutting Plane Algorithm

5.2.1 Motivation

From a graphical perspective, the branch and bound method determines an ILP's solution through a process of repeatedly subdividing the feasible region of the corresponding relaxation LP. The *cutting plane algorithm*, developed by Ralph Gomory, also involves an iterative process of solving LPs. Each new LP consists of its predecessor LP, combined with an additional, cleverly constructed constraint that "trims" the feasible region, so to speak, without removing feasible lattice points.

Our primary focus will be on pure ILPs (as opposed to MILPs) having the property that all constraint bounds and all decision variable coefficients are integer-valued. The GLKC ILP,

$$\begin{aligned} &\text{maximize } z = 3x_1 + 4x_2 \\ &\quad \text{subject to} \\ &\qquad 2x_1 + x_2 \leq 6 \\ &\qquad 2x_1 + 3x_2 \leq 9 \\ &\qquad x_1, x_2 \geq 0; x_1, x_2 \in \mathbb{Z}, \end{aligned} \tag{5.18}$$

is a simple example of such a problem, and we will use it as a context for constructing our new technique.

5.2.2 The Algorithm

We begin by adding slack variables, s_1 and s_2, to the constraints in (5.18) and performing the simplex algorithm on the relaxation LP. The resulting final tableau is shown in Table 5.5. Note that none of the basic variables are integer-valued.

TABLE 5.5: Final tableau of the GLKC ILP relaxation

z	x_1	x_2	s_1	s_2	RHS
1	0	0	$\frac{1}{4}$	$\frac{5}{4}$	$\frac{51}{4}$
0	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{9}{4}$
0	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$

An important tool in subsequent steps is the *integer component*, or *floor function*

of a real number. If x belongs to $\mathbb{R}$, then the integer component of x, denoted by $\lfloor x \rfloor$, is the largest integer less than or equal to x. For example, $\left\lfloor \frac{9}{4} \right\rfloor = 2$ and $\left\lfloor -\frac{1}{4} \right\rfloor = -1$. Clearly, $\lfloor x \rfloor = x$ for every integer x. The *fractional part* of x is defined as the difference $x_f = x - \lfloor x \rfloor$, which satisfies $0 \leq x_f < 1$.

We now choose a row in (5.5) for which the corresponding basic variable is not integer-valued. In this particular case, any row will suffice, but as a general rule, we choose the row whose basic variable has its fractional value closest to one-half. Thus, in this case we select the bottom row, which corresponds to the equation

$$x_2 - \frac{1}{2}s_1 + \frac{1}{2}s_2 = \frac{3}{2}. \tag{5.19}$$

Each variable coefficient in (5.19) can be uniquely expressed as the sum of integer and fractional components so that the equation is the same as

$$(1+0)x_2 + \left(-1 + \frac{1}{2}\right)s_1 + \left(0 + \frac{1}{2}\right)s_2 = 1 + \frac{1}{2}. \tag{5.20}$$

Rewriting (5.20) so that coefficients having integer components and coefficients having fractional components are grouped on opposite sides of the equal sign, we obtain

$$x_2 - s_1 - 1 = \frac{1}{2} - \frac{1}{2}s_1 - \frac{1}{2}s_2. \tag{5.21}$$

The "trimming" of the relaxation LP's feasible region arises from the fact that whenever terms of an equation corresponding to a row of the relaxation's final tableau are separated in this manner, with terms having integer component coefficients on one side of the equation and terms having fractional component coefficients on the other, each side of the resulting equation is nonpositive whenever (x_1, x_2) is a feasible solution of the ILP.

To demonstrate that this assertion holds for this particular example, we first consider the constraint equations, associated with the original relaxation of (5.18):

$$\begin{aligned} 2x_1 + x_2 + s_1 &= 6 \\ 2x_1 + 3x_2 + s_2 &= 9. \end{aligned}$$

If (x_1, x_2) is a feasible solution of the ILP, then the corresponding values of s_1 and s_2 must be integer-valued. This fact follows from the feasibility assumption, meaning x_1 and x_2 are both integer-valued, along with our assumption that all constraint bounds and all decision variable coefficients of the ILP are integer-valued. Therefore, $x_2 - s_1 - 1$ is integer-valued, and, in light of (5.21),

so is $\frac{1}{2} - \frac{1}{2}s_1 - \frac{1}{2}s_2$. But both s_1 and s_2 are nonnegative by feasibility so that $\frac{1}{2} - \frac{1}{2}s_1 - \frac{1}{2}s_2 \leq \frac{1}{2}$. Since $\frac{1}{2} - \frac{1}{2}s_1 - \frac{1}{2}s_2$ is an integer bounded above by $\frac{1}{2}$, it must be nonnegative. By (5.21) $x_2 - s_1 - 1 \leq 0$ as well. Thus, we have established that if (x_1, x_2) is feasible solution of the ILP, then the inequalities $x_2 - s_1 - 1 \leq 0$ and $\frac{1}{2} - \frac{1}{2}s_1 - \frac{1}{2}s_2 \leq 0$ are both valid. This means that adding one or both of them to the original ILP will "trim" the feasible region, so to speak, without removing from consideration feasible solutions of the ILP.

We now introduce a new constraint to the relaxation that incorporates the slack-variable inequality, $\frac{1}{2} - \frac{1}{2}s_1 - \frac{1}{2}s_2 \leq 0$. If s_3 denotes a new, nonnegative slack variable and if we rearrange terms, this inequality leads to the equation

$$-\frac{1}{2}s_1 - \frac{1}{2}s_2 + s_3 = -\frac{1}{2}. \tag{5.22}$$

We now add this equation to the relaxation solution tableau, (5.5), by adding a new row and column. The result is shown in Table 5.6.

TABLE 5.6: Tableau for the relaxation after a new slack variable, s_3, has been introduced

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	$\frac{1}{4}$	$\frac{5}{4}$	0	$\frac{51}{4}$
0	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	$\frac{9}{4}$
0	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{3}{2}$
0	0	0	$-\frac{1}{2}$	$-\frac{1}{2}$	1	$-\frac{1}{2}$

At this stage of the algorithm, both s_1 and s_2 are nonbasic and $s_3 = -\frac{1}{2}$. Thus, the current basic variables constitute a basic, but not basic feasible solution, of the LP formed by adding constraint (5.22) to the relaxation of (5.18). To obtain a basic feasible solution, we utilize the dual simplex algorithm from Section 4.3.

Since only s_3 is a negative basic variable, we apply the ratio test to the bottom row of (5.6) The result is that variable s_1 replaces s_3 as a basic variable. The resulting tableau is given by (5.7).

The tableau in (5.7) marks the end of the first iteration of what we refer to as the *cutting plane algorithm*.

The decision variable x_1 is not integer-valued, so we repeat the process used in the first iteration and determine a second additional constraint, which

TABLE 5.7: Tableau after the first iteration of the cutting plane algorithm

z	x_1	x_2	s_1	s_2	s_3	RHS
1	0	0	0	1	$\frac{1}{2}$	$\frac{25}{2}$
0	1	0	0	-1	$\frac{3}{2}$	$\frac{3}{2}$
0	0	1	0	1	-1	2
0	0	0	1	1	-2	1

is added to the LP relaxation. We first select the row containing the basic variable whose fractional part is closest to $\frac{1}{2}$, which in this case corresponds to x_1.

While we could repeat the entire process of recording the equation corresponding to this row, decomposing variable coefficients into their integer and fractional components, and grouping terms with fractional coefficients on one side of the equation, we see from close inspection of the first iteration that a simpler means exists. Namely, we may perform the following steps:

1. Compute the fractional component of each entry in the row corresponding to x_1. The resulting values yield $\left[0, 0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}\right]$.

2. Add to (5.7) a new slack variable, s_4. Entries in its corresponding column are all zero, except in a new row added to the tableau where the entry corresponding to s_4 equals 1 and all other entries consist of additive inverses of the previously computed fractional components. The resulting tableau is given by (5.8).

3. Perform the dual simplex algorithm. The first step of this algorithm focuses on the new row just added to the tableau, which corresponds to the negative basic variable s_4.

TABLE 5.8: Tableau for the relaxation after a new slack variable, s_4, has been introduced

z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
1	0	0	0	1	$\frac{1}{2}$	0	$\frac{25}{2}$
0	1	0	0	-1	$\frac{3}{2}$	0	$\frac{3}{2}$
0	0	1	0	1	-1	0	2
0	0	0	1	1	-2	0	1
0	0	0	0	0	$-\frac{1}{2}$	1	$-\frac{1}{2}$

Executing the dual simplex algorithm a second time, we see that s_3 replaces s_4 as a basic variable. The resulting pivot leads to the result in Table (5.9).

TABLE 5.9: Tableau for the GLKC ILP after second iteration of cutting plane algorithm

z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
1	0	0	0	1	0	1	12
0	1	0	0	-1	0	3	0
0	0	1	0	1	0	-2	3
0	0	0	1	1	0	-4	3
0	0	0	0	0	1	-2	1

The tableau indicates an optimal solution to original ILP given by $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ and $z = 12$. In addition, all slack variables are integer-valued.

5.2.3 A Step-by-Step Maple Implementation of the Cutting Plane Algorithm

The cutting plane algorithm is easy to implement with Maple. What follows is a portion of a worksheet, **Cutting Plane Algorithm.mw**, demonstrating how this is accomplished for the GLKC ILP. Missing from the beginning of the worksheet are the following:

1. Contents of the worksheet **Simplex Algorithm.mw** from Section 2.1, which allows the user to solve the ILP's relaxation
2. The `ColumnRatios` procedure found at the end of Section 4.3.

In addition, we have omitted from the following worksheet a new Maple procedure, called `fracpart`, which computes the fractional part of its argument. It can be defined at the start of the worksheet as follows:

```
fracpart:=proc(x) local y, y=x-floor(x):RETURN((y)):end:
```

For example, `fracpart(-.25)` returns .75.

We begin this portion of the worksheet with the final tableau matrix, `LPMatrix`, that results when the simplex algorithm is used to solve the GLKC's ILP relaxation. Recall again our numbering convention that column 0 and row 0 denote the leftmost column and top row, respectively, of the tableau matrix.

```
> Tableau(LPMatrix);
  # Display final tableau for GLKC ILP relaxation.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & RHS \\ 1 & 0 & 0 & \frac{1}{4} & \frac{5}{4} & \frac{51}{4} \\ 0 & 1 & 0 & \frac{3}{4} & -\frac{1}{4} & \frac{9}{4} \\ 0 & 0 & 1 & -\frac{1}{2} & \frac{1}{2} & \frac{3}{2} \end{bmatrix}$$

```
> evalf(%);
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & RHS \\ 1 & 0 & 0 & .25 & 1.25 & 12.75 \\ 0 & 1 & 0 & .75 & -.25 & 2.25 \\ 0 & 0 & 1 & -.5 & .5 & 1.5 \end{bmatrix}$$

```
> m:=m+1:
> for j from 2 to m do evalf(fracpart(LPMatrix[j,n+m+1]));end do;
  # Determine basic variable whose fractional part is closest to
  one half.
```

$$.25$$
$$.50$$

```
> k:=2:
  # Basic variable corresponding to row 2 has fractional part closest
  to one half.
> NewRow:=-Vector[row]([seq(-fracpart(RowCoefficients[j]),j=1..(n+m+1)),-1,
  -fracpart(LPMatrix[k+1,n+m+1])]);
  # Create new bottom row corresponding to added constraint.
```

$$NewRow := \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

```
> LPMatrix:=<<SubMatrix(LPMatrix,1..m,1..(n+m))|ZeroVector(m)|
  SubMatrix(LPMatrix,1..m,(m+n+1)..(m+n+1))>,NewRow>;
  # Create new matrix incorporating addition of new slack variable
  to original LP's relaxation.
```

$$LPMatrix := \begin{bmatrix} 1 & 0 & 0 & \frac{1}{4} & \frac{5}{4} & 0 & \frac{51}{4} \\ 0 & 1 & 0 & \frac{3}{4} & -\frac{1}{4} & 0 & \frac{9}{4} \\ 0 & 0 & 1 & -\frac{1}{2} & \frac{1}{2} & 0 & \frac{3}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

```
> x:=array(1..n):s:=array(1..m):
  # Create arrays of decision and slack variables.
> Labels:=Matrix(1,2+n+m,[z,seq(x[i],i=1..n),seq(s[j],j=1..m),RHS]):
  # Create a new top row of labels.
> Tableau(LPMatrix);
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & 0 & 0 & \frac{1}{4} & \frac{5}{4} & 0 & \frac{51}{4} \\ 0 & 1 & 0 & \frac{3}{4} & -\frac{1}{4} & 0 & \frac{9}{4} \\ 0 & 0 & 1 & -\frac{1}{2} & \frac{1}{2} & 0 & \frac{3}{2} \\ 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

```
> ColumnRatios(LPMatrix,m);
  # Perform dual simplex algorithm ratio test on row m.
```

"Column 1 Ratio Undefined"
"Column 2 Ratio Undefined"
"Column 3 Ratio = -.5000000000"
"Column 4 Ratio =-2.5000000000"
"Column 5 Ratio =0."

```
> Iterate(LPMatrix,m,3);
  # Pivot on entry in row m, column 3 so that s1 replaces s3 as
  a basic variable.
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & RHS \\ 1 & 0 & 0 & 0 & 1 & \frac{1}{2} & \frac{25}{2} \\ 0 & 1 & 0 & 0 & -1 & \frac{3}{2} & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 1 & -1 & 2 \\ 0 & 0 & 0 & 1 & 1 & -2 & 1 \end{bmatrix}$$

```
> m:=m+1: # End of first iteration; repeat process.
> for j from 2 to m do evalf(fracpart(LPMatrix[j,n+m+1]));end do;
  # Determine basic variable whose fractional part is closest to
  one half.
```

.5
0
0

```
> k:=1:
  # Basic variable corresponding to row 1 has fractional part closest
  to one half.
> NewRow:=-Vector[row]([seq(fracpart(LPMatrix[k+1,j]),j=1..(n+m)),-1,
  fracpart(LPMatrix[k+1,n+m+1])]);
  # Create new bottom row corresponding to added constraint.
```

$$NewRow := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

```
> LPMatrix:=<<SubMatrix(LPMatrix,1..m,1..(n+m))|ZeroVector(m)|
  SubMatrix(LPMatrix,1..m,(m+n+1)..(m+n+1))>,NewRow>;
  # Create new matrix incorporating addition of new slack variable
  to original LP's relaxation.
> x:=array(1..n):s:=array(1..m):
  # Create arrays of decision and slack variables.
> Labels:=Matrix(1,2+n+m,[z,seq(x[i],i=1..n),seq(s[j],j=1..m),RHS]):
  # Create a new top row of labels.
```

```
> Tableau(LPMatrix);
```

$$\begin{bmatrix} z & x_1 & x_2 & s_1 & s_2 & s_3 & s_4 & RHS \\ 1 & 0 & 0 & 0 & 1 & \frac{1}{2} & 0 & \frac{25}{2} \\ 0 & 1 & 0 & 0 & -1 & \frac{3}{2} & 0 & \frac{3}{2} \\ 0 & 0 & 1 & 0 & 1 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & -2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$$

```
> ColumnRatios(LPMatrixNew,m); # Perform ratio test on row m.
```

"Column 1 Ratio Undefined"
"Column 2 Ratio Undefined"
"Column 3 Ratio Undefined"
"Column 4 Ratio Undefined"
"Column 5 Ratio = -1."
"Column 6 Ratio = 0."

```
> Iterate(LPMatrix,m,5);
  # Pivot on entry in row m, column 5 to obtain final tableau.
```

$$\begin{bmatrix} z & x1 & x2 & s1 & s2 & s3 & s4 & RHS \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 12 \\ 0 & 1 & 0 & 0 & -1 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 0 & -4 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix}$$

5.2.4 Comparison with the Branch and Bound Method

When applied to the GLKC ILP, the cutting plane algorithm required only two iterations starting with the relaxation LP's final tableau, whereas the branch and bound algorithm required four branching levels. On the basis of this isolated example, we might conclude that the cutting plane algorithm is the more efficient of the two algorithms. However, as a general rule, the opposite is true, and the branch and bound is the preferred technique. Contemporary methods utilize what are known as "cut and branch" techniques, whereby the cutting plane algorithm is used to help solve intermediate LPs that arise within a larger branch and bound framework.

Exercises Section 5.2

Use the Cutting Plane Algorithm to solve each of the following ILPs.

1.

$$\begin{aligned} &\text{maximize } z = 2x_1 + 3x_2 \\ &\text{subject to} \\ &4x_1 + 3x_2 \leq 8 \\ &2x_1 + x_2 \leq 15 \\ &x_1 + 2x_2 \leq 7 \\ &x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z} \end{aligned}$$

2.

$$\begin{aligned} &\text{maximize } z = 4x_1 + 3x_2 + 5x_3 \\ &\text{subject to} \\ &x_1 + x_2 + x_3 \leq 17 \\ &6x_1 + 3x_2 + 3x_3 \leq 22 \\ &3x_1 + 3x_2 \leq 13 \\ &x_1, x_2, x_3 \geq 0, x_1, x_2, x_3 \in \mathbb{Z} \end{aligned}$$

Part II

Nonlinear Programming

Chapter 6

Algebraic Methods for Unconstrained Problems

6.1 Nonlinear Programming: An Overview

The world we live in is "nonlinear" in that mathematical relationships between quantities are rarely described by simple linear functions. As a result, nonlinear programming is an essential tool for solving problems from a variety of disciplines. The goal of subsequent chapters is to become acquainted with several such problems and to develop an understanding of fundamental mathematical tools used to address them.

6.1.1 The General Nonlinear Programming Model

A nonlinear program, or NLP, is a generalization of the LP in which the objective function is permitted to be nonlinear and constraints are permitted to take the form of nonlinear inequalities. Specifically, an NLP can be expressed as follows:

$$\begin{aligned} &\text{minimize } f(x_1, x_2, \ldots, x_n) \\ &\qquad\qquad \text{subject to} \\ &g_1(x_1, x_2, \ldots, x_n) \leq 0 \\ &g_2(x_1, x_2, \ldots, x_n) \leq 0 \\ &\qquad\vdots \qquad\qquad \vdots \\ &g_m(x_1, x_2, \ldots, x_n) \leq 0. \end{aligned} \tag{6.1}$$

Unless stated otherwise, we will incorporate any sign restrictions into the constraints. Of course NLPs whose goal involves maximization and/or whose constraints include equations as opposed to inequalities, can easily be converted to the form of (6.1) through simple algebraic manipulations. Finally, an NLP having no constraints is said to be *unconstrained*.

6.1.2 Plotting Feasible Regions and Solving NLPs with Maple

Feasible regions for NLPs are a bit more complicated to graph with Maple than are those for LPs, due to the fact Maple's `inequal` command only plots feasible regions corresponding to lists of linear inequalities. However, with a bit of creativity, we can work around this obstacle.

To illustrate this idea, we consider the simple two-variable NLP given by

$$\begin{aligned} \text{maximize} & f(x_1, x_2) = x_1 x_2 \\ & \text{subject to} \\ & x_1^2 + x_2^2 \leq 1 \\ & x_1 \geq 0. \end{aligned} \tag{6.2}$$

Using the notation of (6.1) for the constraints, we note that (6.2) is equivalent to

$$\begin{aligned} \text{maximize} & f(x_1, x_2) = x_1 x_2 \\ & \text{subject to} \\ & x_1^2 + x_2^2 - 1 \leq 0 \\ & -x_1 \leq 0. \end{aligned} \tag{6.3}$$

The first constraint is nonlinear, so Maple's `inequal` command cannot be used to plot the feasible region. Instead, we combine `piecewise` and `implicitplot` as follows:

```
> restart:with(plots):
> g1:=(x1,x2)->x1^2 + x2^2-1;
  # Enter first constraint function.
```

$$g_1 := (x_1, x_2) \to x_1^2 + x_2^2 - 1$$

```
> g2:=(x1,x2)->-x1;
  # Enter second constraint function.
```

$$g_2 := (x_1, x_2) \to -x_1$$

```
> F:=piecewise(g1(x1,x2)<=0 and g2(x1,x2)<=0,1);
  # Define a two-variable function that equals 1 precisely if its
  input is feasible for the NLP.
```

$$F := (x_1, x_2) \to \mathit{piecewise}(g_1(x_1, x_2) \leq 0 \text{ and } g_2(x_1, x_2) \leq 0, 1)$$

```
> FeasibleRegion:=implicitplot(F(x1,x2)=1,x1=0..1,x2=-1..1,color=grey,
  grid=[200,200]):
  # Create feasible region plot using implicitplot command. The
  horizontal and vertical viewing windows are divided into 200
  grid points.
```

```
> display(FeasibleRegion);
```

The result of the `display` command is shown in Figure 6.1.

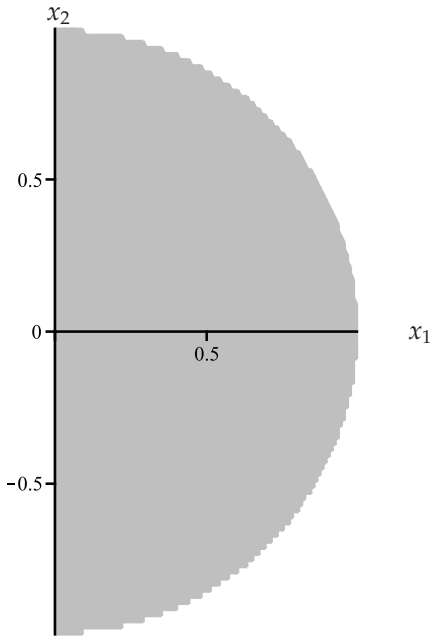


FIGURE 6.1: Feasible region for NLP (6.2).

As was the case in the linear programming setting, contours are useful for estimating the solution of an NLP. In Maple, they can be generated and then superimposed on the feasible region by combining the `contourplot` and `display` commands as was done in Section 1.2. Unfortunately, Maple does not label contours. This fact, along with the nonlinearity of the objective and constraints, can make estimating the solution of an NLP using its contour diagram extremely difficult.

One means of gaining a better sense of how the objective function changes within the feasible region is to add within the `implicitplot` command the options `filled = true, coloring = [white, black]`. Doing so has the effect of shading between contours in such a way that smallest function values correspond to white regions, largest values correspond to black, and intermediate values to varying shades of grey. Maple syntax that achieves this outcome is given as follows:

```
> f:=(x1,x2) -> x1*x2;
```

$$f := (x_1, x_2) \rightarrow x_1 x_2$$

```
> ObjectiveContours:=contourplot(f(x1,x2),x1=0..1,x2=-1..1,filled
  = true, coloring = [white, black]):
  # Create contour plot.
> display({FeasibleRegion,ObjectiveContours});
  # Superimpose contours on previously constructed feasible region.
```

The result in this case produces the graph in Figure 6.2. It suggests that the solution of (6.2) occurs at the point on the unit circle corresponding to $\frac{\pi}{4}$, which is of course $(x_1, x_2) = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$.

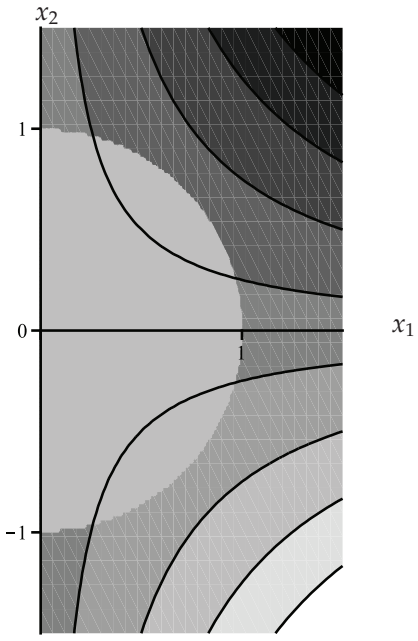


FIGURE 6.2: Feasible region and contour shading for NLP (6.2).

Maple's `NLPSolve` command, located in the `Optimization` package, functions almost exactly as its linear programming counterpart, `LPSolve`. To solve NLP (6.3), we merely enter the following:

```
> restart: with(Optimization):
```

```
> f:=(x1,x2)->x1*x2;
```

$$f := (x_1, x_2) \to x_1 x_2$$

```
> g1:=(x1,x2)->x1^2 + x2^2-1;
```

$$g_1 := (x_1, x_2) \to x_1^2 + x_2^2 - 1$$

```
> g2:=(x1,x2)->-x1;
```

$$g_2 := (x_1, x_2) \to -x_1$$

```
> NLPSolve(f(x1,x2),[g1(x1,x2)<=0,g2(x1,x2)<=0],'maximize');
```

$$.500000000001127654, [x_1 = .707106781187344936, x_2 = .707106781187344936]$$

Thus NLP (6.3) has a solution of $(x_1, x_2) \approx (.7071, .7071)$, with corresponding objective value of .5. In Section 6.4 we will develop algebraic techniques that prove this solution equals $(x_1, x_2) = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$.

6.1.3 A Prototype NLP Example

In Part I, the *FuelPro Petroleum Company*model served to illustrate a variety of important linear programming principles. As two-variable models are convenient for visualizing important concepts in general, we focus in Part II on a simple two-variable NLP.

ConPro Manufacturing Company is a small business that produces large concrete pipes. The company seeks to maximize its profit subject to certain constraints. Assume pipe production is a function of material and labor and that the number of units of each of these quantities is denoted by x_1 and x_2, respectively. The company has determined that production, measured in units of pipe, is given by the product power function

$$P(x_1, x_2) = x_1^{\frac{1}{2}} x_2^{\frac{1}{3}}. \tag{6.4}$$

The function P is known as a *Cobb-Douglas Production Function*, whose general form is given by $P(x_1, x_2) = x_1^{\alpha} x_2^{\beta}$, where $\alpha, \beta > 0$.

Each unit of produced pipe generates \$1400 of revenue but costs \$350 in materials and \$200 in labor to produce. The company has \$40,000 of available funds to spend, and the ratio of labor units to material units must be at least one-third in order to ensure adequate labor to produce pipe from acquired materials. Assume *ConPro* sells all the pipe that it produces.

ConPro 's profit as a function of material and labor is given by revenue, less material and labor cost:

$$\begin{aligned} f(x_1, x_2) &= 1400P(x_1, x_2) - 350x_1 - 200x_2 \\ &= 1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} - 350x_1 - 200x_2. \end{aligned} \tag{6.5}$$

Funding limits imply that $350x_1 + 200x_2 \leq 40,000$, and the requirement of adequate labor to ensure pipe production of acquired materials forces $x_1 \leq 3x_2$. Assuming positive decision variable values capable of taking on non-integer values, we obtain NLP (6.6):`convert`

$$\begin{aligned} \text{maximize} f(x_1, x_2) &= 1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} - 350x_1 - 200x_2 \\ \text{subject to} & \\ 350x_1 + 200x_2 &\leq 40000 \\ x_1 - 3x_2 &\leq 0 \\ x_1, x_2 &> 0. \end{aligned} \tag{6.6}$$

Note that this NLP includes sign restrictions on its decision variables. By our convention, these restrictions contribute two constraints, $-x_1 \leq 0$ and $-x_2 \leq 0$, so that the NLP has four constraints total.

In subsequent chapters, we learn how nonlinear programming tools can be used to address a variety of questions. Among these are the following:

1. How closely does P portray the actual production values?

2. In the absence of any constraints, what combination of material units and labor units maximizes profit? Is this combination necessarily unique?

3. If constraints are incorporated into the model, what new combination maximizes profit? Again, is this combination unique?

4. If exact answers to the previous two questions are unattainable using algebraic means, do numeric methods exist that achieve approximate results?

Answers to these questions incorporate a variety of mathematical tools, most notably those from linear algebra and multivariate calculus.

Waypoint 6.1.1. Consider the *ConPro* NLP, (6.6).

1. Graph the feasible region for this NLP.
2. Use Maple to create a contour diagram of the objective function and estimate the NLP's solution.
3. Solve the NLP using Maple's `NLPSolve` command.

Exercises Section 6.1

1. Express each of the following NLPs in the general form (6.1). Then create a contour diagram for the objective function, and use your result to estimate the solution of the NLP. Check your estimate using Maple's `NLPSolve` command.

   (a)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1^2 - 6x_1 + x_2^2 - 4x_2 \\ \text{subject to} & \\ x_1 + x_2 &\leq 3 \\ x_1 &\leq 1 \\ x_2 &\geq 0 \end{aligned}$$

   (b)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1^3 - 2x_1 - x_2 \\ \text{subject to} & \\ x_1 + x_2 &\leq 1 \\ x_1, x_2 &\geq 0 \end{aligned}$$

   (c)

$$\begin{aligned} \text{maximize } & f(x_1, x_2) = x_1^2 x_2 - x_1 \\ \text{subject to} & \\ -x_1 + x_2 &\geq 1 \\ x_1^2 + x_2^2 &\leq 2 \end{aligned}$$

(d)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = 4x_1^2 - 12x_1 + x_2^2 - 6x_2 \\ & \text{subject to} \\ & x_1 + x_2^2 \leq 2 \\ & 2x_1 + x_2 = 1 \end{aligned}$$

(e)

$$\begin{aligned} \text{maximize } & f(x_1, x_2) = \ln(x_2 + 1) - x_1^2 \\ & \text{subject to} \\ & (x_1 - 1)^2 + (x_2 - 2)^2 \leq 1 \\ & -2x_1 + x_2 \leq 1 \end{aligned}$$

(f)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = \frac{x_1 + 2}{x_2 + 1} \\ & \text{subject to} \\ & x_1 + 2x_2 \leq 3 \\ & -x_1 + x_2 \leq 1 \end{aligned}$$

2. (*Pam's Pentathlon Training Program*) The women's pentathlon consists of five events: the 800 meter run, the 60 meter hurdles, the high jump and long jump, and the shot put throw.[1] Pam has decided to begin an off-season training program combining weight lifting, distance running, and speed workouts (e.g., sprints) in order to improve her performance in her weakest two of the five events: the 800 meter run and the shot put. Each week Pam plans to devote a certain number of hours to each of these three activities, with the number devoted to each remaining constant from one week to another.

   Currently, Pam completes the 800 meter run in 3 minutes and throws the shot put 6.7 meters. Pam's coach estimates that for each hour per week Pam devotes to weight lifting, she will decrease her 800 meter run time by half that many seconds and increase her shot put distance by one-tenth of a meter. Thus, for example, lifting weights two hours per week will decrease her 800 meter run time by 1 second and increase her shot put distance by .2 meters. Similarly, the coach estimates that for each hour per week Pam devotes to distance running and for each hour per week Pam devotes to speed workouts, she will decrease her 800 meter run time by that many seconds.

[1] Based upon Ladany, [20], (1975).

Pam's workout routine must adhere to certain constraints, which are as follows:

(a) Pam is expected to train between six and ten hours per week.

(b) Between two and four hours of this time must be devoted to weight lifting.

(c) Between three and four hours should be spent distance running.

(d) In order to ensure Pam devotes sufficient time to speed workouts, she must allot at least 60% of her total running time to this activity.

According to International Amateur Athletic Federation guidelines, the number of points contributing to Pam's total pentathlon score by her 800 meter run performance is determined using the formula, $P_1 = .11193(254 - t)^{1.88}$, where t is the time, in seconds, required for her to complete the run. Similarly, the number of points contributing to Pam's total pentathlon score by her shot put performance equals $P_2 = 56.0211(d - 1.5)^{1.05}$, where d is the distance, in meters, of her throw.

Based upon this information, set up and solve an NLP having three decision variables and ten constraints (including sign restrictions), whose solution indicates the number of hours Pam should devote weekly to weight lifting, distance running, and speed workouts in order to maximize her score in the 800 meter run and shot put components of the pentathlon. By how much will her total score increase?

6.2 Differentiability and a Necessary First-Order Condition

We begin our study of nonlinear programming by focusing on unconstrained NLPs. If $S \subseteq \mathbb{R}^n$ and the objective function $f : S \to \mathbb{R}$, then the general form of such a problem is given by

$$\text{minimize (or maximize) } f(\mathbf{x}), \text{ where } \mathbf{x} \in S. \tag{6.7}$$

We shall later discover that constrained NLPs are solved using a method that involves cleverly converting them to an unconstrained problem of the form (6.7). For unconstrained NLPs, our investigation follows a line of reasoning similar to that from a single-variable calculus course. In this section we derive necessary conditions for a feasible point to be an optimal solution. This step will be the easy part. The more challenging task, which is addressed in Sections 6.3 and 6.4, is to establish sufficient conditions.

6.2.1 Differentiability

Deriving necessary conditions begins with stating a clear definition of differentiability. Throughout subsequent discussions, we let $\|\mathbf{x}\|$ denote the usual Euclidean vector norm in $\mathbb{R}^n$.

Definition 6.2.1. Suppose that $S \subseteq \mathbb{R}^n$ and that $f : S \to \mathbb{R}$. We say that the function f is *differentiable* at $\mathbf{x}_0$ in S if and only if there exists a vector depending upon f and $\mathbf{x}_0$, called *the gradient of f at* $\mathbf{x}_0$, written $\nabla f(\mathbf{x}_0)$, and a scalar function $R(\mathbf{x}_0; \mathbf{x})$ dependent upon $\mathbf{x}_0$ and defined for all $\mathbf{x}$ in S, such that

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) + \|\mathbf{x} - \mathbf{x}_0\| R(\mathbf{x}_0; \mathbf{x}), \tag{6.8}$$

for all $\mathbf{x}$ in S and $\lim_{\mathbf{x} \to \mathbf{x}_0} R(\mathbf{x}_0; \mathbf{x}) = 0$.

Here are some important facts to keep in mind regarding this definition.

1. Since $\nabla f(\mathbf{x}_0)$ is a column vector, its transpose is a row vector in $\mathbb{R}^n$. Thus, $\nabla f(x_0)^t(\mathbf{x} - \mathbf{x}_0)$ is a scalar, as are the remaining three terms in (6.8).

2. Equation (6.8), due to the presence of the limit, implicitly assumes that f is defined at inputs sufficiently close to $\mathbf{x}_0$. Consequently, there must exist a small neighborhood, or open disk, about $\mathbf{x}_0$ that is contained in S. A set $S \subseteq \mathbb{R}^n$ in which every point has an open disk about it contained in S, is said to be *open*.

3. If the last term is discarded from (6.8), we obtain the function

$$T(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0), \tag{6.9}$$

the linearization, or linear approximation of f, based at $\mathbf{x}_0$, which is analogous to the tangent line formula from calculus. Thus, $\|\mathbf{x}-\mathbf{x}_0\|R(\mathbf{x}_0;\mathbf{x})$ is an error term in this approximation, and the condition that $\lim_{\mathbf{x}\to\mathbf{x}_0} R(\mathbf{x}_0;\mathbf{x}) = 0$ indicates how quickly this error term decays to zero as $\mathbf{x} \to \mathbf{x}_0$.

An elementary fact, whose proof we omit, is that the components of the gradient consist of the first-order partial derivatives of f. That is,

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}, \tag{6.10}$$

where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$. This result is easily proven using (6.8). If each partial derivative, $\dfrac{\partial f}{\partial x_i}$, where $1 \le i \le n$, is continuous on S, we say that $\nabla f(\mathbf{x})$ is *continuously differentiable* on S.

In Maple, the `Gradient` command, located in the `VectorCalculus` package provides a means for computing the gradient. Its general form is given by `Gradient(`*`expression`*`,`*`variable list`*`)`. Here, *`variable list`* indicates the variables with respect to which the gradient of *`expression`* is computed. The output of the command is an expression given in terms of unit vectors corresponding to each variable. Each unit vector takes the form $\mathbf{e}_{variable}$. A simple example of the command's usage is as follows:

```
> with(VectorCalculus):
> Gradient(x1^2 +x1*x2^3,[x1,x2]);
```

$$(2x_1 + x_2^3)\bar{e}_{x1} + 3x_1x_2^2\bar{e}_{x2}$$

If f is a function, the `unapply` command can be used to define the corresponding gradient function. For example, if $f(x_1, x_2) = x_1^2 + x_2^2$, its gradient function is constructed using the following syntax. Here we name this function `Delf`.

```
> with(VectorCalculus):
> f:=(x1,x2)->x1^2+x2^2;
```

$$f := (x_1, x_2) \to x_1^2 + x_2^2$$

```
> Delf:=unapply(Gradient(f(x1,x2),[x1,x2]),[x1,x2]):
> Delf(x1,x2);
```

$$2x_1\mathbf{e}_{x_1} + 2x_2\mathbf{e}_{x_2}$$

Using Definition (6.2.1) to establish a function is differentiable at an input, $\mathbf{x}_0$, can be somewhat tedious due to the fact we must show $\lim_{\mathbf{x}\to\mathbf{x}_0} R(\mathbf{x}_0;\mathbf{x}) = 0$. The following Waypoint illustrates one means of accomplishing this.

◆ ◆

Waypoint 6.2.1. Consider the function $f(x_1, x_2) = x_1x_2^2 - 2x_1^2 + 3x_2$ along with $\mathbf{x}_0 = \begin{bmatrix}2\\1\end{bmatrix}$.

1. Calculate the linear approximation

$$T(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0),$$

where $\mathbf{x} = \begin{bmatrix}x_1\\x_2\end{bmatrix}$. Use your result to determine an explicit representation, in terms of x_1 and x_2, for the remainder term, $R(\mathbf{x}_0;\mathbf{x})$.

2. Now express $R(\mathbf{x}_0;\mathbf{x})$ in polar form as a function of $r > 0$ and θ, where $x_1 = 2 + r\cos(\theta)$ and $x_2 = 1 + r\sin(\theta)$. Show that $|R(\mathbf{x}_0;\mathbf{x})| \le C \cdot r$, where C is a constant. Since $r = \|\mathbf{x} - \mathbf{x}_0\|$, this inequality proves that $\lim_{\mathbf{x}\to\mathbf{x}_0} R(\mathbf{x}_0;\mathbf{x}) = 0$.

◆ ◆

6.2.2 Necessary Conditions for Local Maxima or Minima

In the linear setting, we saw that for any LP, one of four possible outcomes must occur: the LP has a unique solution, has infinitely many solutions, is infeasible, or is unbounded. For the general unconstrained NLP (6.7), other cases are possible, e.g., exactly two solutions. For this reason, we must exercise great care when defining the concept of maximum and minimum objective value in the nonlinear setting.

Definition 6.2.2. Suppose $S \subseteq \mathbb{R}^n$ and $f : S \to \mathbb{R}$. We say that $\mathbf{x}_0$ is a *local minimum* of f if there is a sufficiently small $\epsilon > 0$ such that

$$f(\mathbf{x}_0) \le f(\mathbf{x}) \text{ for all } \mathbf{x} \text{ in } S \text{ satisfying } \|\mathbf{x} - \mathbf{x}_0\| < \epsilon.$$

If in fact $f(\mathbf{x}_0) \le f(\mathbf{x})$ for all $\mathbf{x}$ in S, we say that $\mathbf{x} = \mathbf{x}_0$ is a *global minimum* of f on S.

Naturally, if the inequalities are reversed, "minimum" is replaced by "maximum." The term *strict* is also frequently added to describe situations when inequalities are strict. For example, to say that $\mathbf{x} = \mathbf{x}_0$ is a *strict global minimum* means $f(\mathbf{x}_0) < f(\mathbf{x})$ for all $\mathbf{x}$ in S with $\mathbf{x} \neq \mathbf{x}_0$. *Local optimal solution* refers to either a local minimum or maximum when the context of the appropriate term is clear. *Global optimal solution* is defined similarly.

A function can have many local maxima and/or minima, and many inputs can share the distinction of being the global maximum (or minimum).

When $n = 1$, the gradient is nothing more than the derivative of a function of one variable. Recall that in the single-variable setting, the first step towards maximizing or minimizing a differentiable function involves determining roots of the derivative, which we denote as the *critical points*. The same principle, as formally stated by Theorem 6.2.1, applies to the general unconstrained NLP.

Theorem 6.2.1. Suppose $S \subseteq \mathbb{R}^n$ is an open set and that $f : S \to \mathbb{R}$ is differentiable at $\mathbf{x}_0$. If f has a local minimum or maximum at $\mathbf{x}_0$, then $\nabla f(\mathbf{x}_0) = \mathbf{0}$.

Proof. The proof involves little more than applying the single-variable result to each component of $\mathbf{x}_0$. Without loss of generality, assume f has a local minimum at $\mathbf{x}_0$. Fix an arbitrary j, where $1 \leq j \leq n$ and write $\mathbf{x}_0 = \begin{bmatrix} x_{0,1} \\ x_{0,2} \\ \vdots \\ x_{0,j} \\ \vdots \\ x_{0,n} \end{bmatrix}$. Define f_j to be the "cross-section" obtained by fixing all but the jth component of f. In other words,

$$f_j(x) = f\left(\begin{bmatrix} x_{0,1} \\ \vdots \\ x_{0,j-1} \\ x \\ x_{0,j+1} \\ \vdots \\ x_{0,n} \end{bmatrix}\right).$$

Observe that

$$f_j'(x_{0,j}) = \left[\nabla f(\mathbf{x}_0)\right]_j, \tag{6.11}$$

the jth component of $\nabla f(\mathbf{x}_0)$.

Since f has a local minimum at $\mathbf{x}_0$, the function f_j has a local minimum at

$x = x_{0,j}$, and therefore $f_j'(x_{0,j}) = 0$ by the single-variable result. Because j was arbitrary, $[\nabla f(\mathbf{x}_0)]_j = 0$ for all $1 \leq j \leq n$, implying $\nabla f(\mathbf{x}_0) = \mathbf{0}$. □

In future discussions, if $S \subseteq \mathbb{R}^n$ is open and $f : S \to \mathbb{R}$, we will say $\mathbf{x}_0$ in $\mathbb{R}^n$ is a *critical point* of f if $\nabla f(\mathbf{x}_0)$ is zero of undefined. Of course, if f is differentiable on S and $\mathbf{x}_0$ is a critical point of f, then $\nabla f(\mathbf{x}_0) = \mathbf{0}$.

In the single-variable setting, a critical point need not be a local minimum or maximum. The same can be said of the unconstrained NLP insofar as Theorem 6.2.1 merely provides a *necessary first-order condition*, stated in terms of the gradient. If $\nabla f(\mathbf{x}_0) \neq \mathbf{0}$, then $\mathbf{x}_0$ can be neither a local minimum nor a local maximum. In subsequent sections we establish *sufficient conditions*, which are analogous to the first- and second-derivative tests from the single-variable setting and which are adequate to guarantee $\mathbf{x}_0$ is a local minimum or maximum.

◆ ——— ◆

Waypoint 6.2.2. Suppose $f(x_1, x_2) = x_1^3 - 3x_1x_2^2$. A surface plot of this function, which is sometimes called a "monkey saddle," is shown in Figure 6.3.

1. Show that $\mathbf{x}_0 = \mathbf{0}$ is the only critical point of f.
2. Explain why this critical point is neither a local maximum nor a local minimum.

◆ ——— ◆

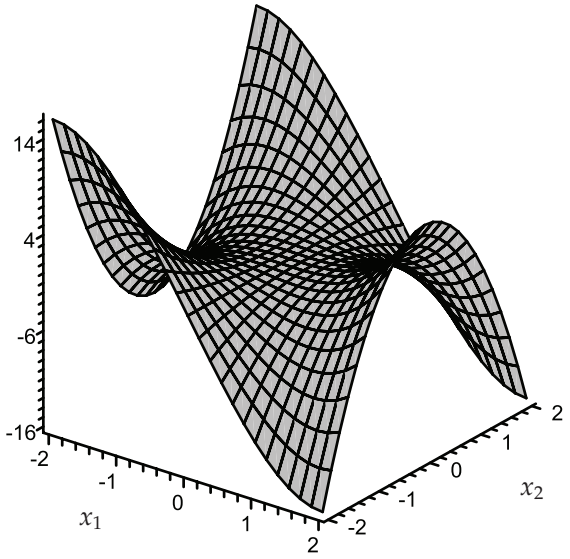


FIGURE 6.3: Surface plot of $f(x_1, x_2) = x_1^3 - 3x_1x_2^2$.

Exercises Section 6.2

1. Compute the gradient of the *ConPro* objective function as a function of $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

2. Use Definition 6.2.1 to show that $f(x_1, x_2) = x_1^3 - 2x_1 - x_2$ is differentiable at $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

3. Suppose $S \subseteq \mathbb{R}^n$, $f : S \to \mathbb{R}$ and $\mathbf{x}_0$ belongs to the interior of S. If $\mathbf{d}$ is a nonzero vector in $\mathbb{R}^n$, we define the *directional derivative of f at $\mathbf{x}_0$ in the direction of $\mathbf{d}$* as

$$f'_{\mathbf{d}}(\mathbf{x}_0) = \lim_{h \to 0^+} \frac{f(\mathbf{x}_0 + h\mathbf{d}) - f(\mathbf{x}_0)}{h}, \tag{6.12}$$

provided this limit exists.

   (a) Show that if f is differentiable at $\mathbf{x}_0$, then the limit in (6.12) exists

and that

$$f'_{\mathbf{d}}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0)^t \mathbf{d}.$$

(b) Show that the preceding quantity $f'_{\mathbf{d}}(\mathbf{x}_0)$ depends only upon the direction of $\mathbf{d}$, not its magnitude. Hence, $\mathbf{d}$ may always be assigned a unit vector value.

(c) For the *ConPro* objective function f, determine the directional derivative of f in the direction of the origin at $(x_1, x_2) = (10, 20)$.

4. Suppose $S \subseteq \mathbb{R}^n$, $f : S \to \mathbb{R}$, and f is differentiable at $\mathbf{x}_1$ in S. Fix $\mathbf{x}_2$ in S. (Note: The differentiability assumption implies $\mathbf{x}_1 + h\mathbf{x}_2$ also belongs to S for h sufficiently close to zero.) Show that $\phi(h) = f(\mathbf{x}_1 + h\mathbf{x}_2)$ is differentiable at $h = 0$ and satisfies

$$\phi'(0) = \nabla f(\mathbf{x}_1)^t \mathbf{x}_2.$$

6.3 Convexity and a Sufficient First-Order Condition

In the single-variable setting, the concavity of a function near a critical point conveys useful information for classifying the critical point as a local maximum or minimum. A simple example is given by

$$f(x) = |x|^{\frac{3}{2}},$$

whose graph is shown in Figure 6.4.

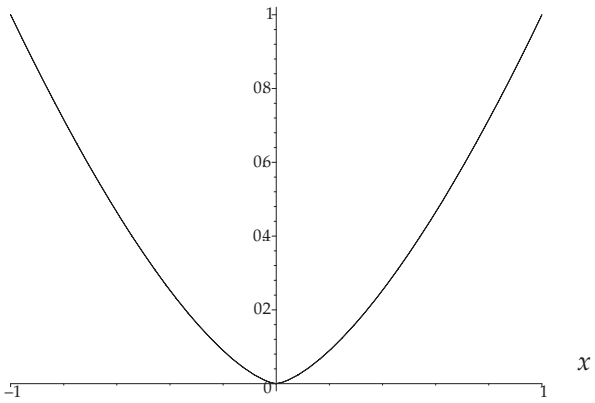


FIGURE 6.4: Graph of $f(x) = |x|^{\frac{3}{2}}$.

This function is differentiable at the origin, where $f'(0) = 0$. However, $f''(0)$ does not exist so that the second derivative test cannot be used to classify $x = 0$ as a minimum. Clearly though, the concavity of the graph indicates this critical point is a minimum.

6.3.1 Convexity

The preceding example illustrates how, in the single-variable setting, concavity still plays a role in classifying a critical point, even though the function may not be twice-differentiable there. As we shall discover, this principle also applies to the general unconstrained NLP. However, instead of using phrases such as "concave up" and "concave down" to describe the behavior of a function, we will instead use the terms *convex* and *concave*. Before defining these terms in the context of functions, we first define convexity for sets.

Definition 6.3.1. Suppose that $S \subseteq \mathbb{R}^n$. We say that S is *convex* if and only if for any $\mathbf{x}_1$ and $\mathbf{x}_2$ in S and for any scalar $t \in [0, 1]$, the linear combination $t\mathbf{x}_1 + (1 - t)\mathbf{x}_2$ also belongs to S.

Such a linear combination, $t\mathbf{x}_1 + (1-t)\mathbf{x}_2$, in which weights are nonnegative and sum to unity, is known as a *convex combination* of $\mathbf{x}_1$ and $\mathbf{x}_2$. Thus, S is convex if, given two arbitrary points, $\mathbf{x}_1$ and $\mathbf{x}_2$ in S, any convex combination of the two points is again in S. If S is contained in $\mathbb{R}^2$ or $\mathbb{R}^3$, this simply means that if $\mathbf{x}_1$ and $\mathbf{x}_2$ belong to S, then so does the line segment connecting them.

We now use convexity of sets to define convexity for functions.

Definition 6.3.2. Suppose that $S \subseteq \mathbb{R}^n$ is convex and that $f : S \to \mathbb{R}$. We say that the function f is *convex on the set* S if and only if for every $\mathbf{x}_1$ and $\mathbf{x}_2$ in S and every $t \in [0,1]$,

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2). \tag{6.13}$$

Stated another way: "The function evaluated at a convex combination of inputs is bounded by the corresponding convex combination of function outputs." If, in the preceding inequality, the direction of the inequality is reversed, we say that f is *concave*.

Observe that the requirement S be convex is needed in order to guarantee that $t\mathbf{x}_1 + (1-t)\mathbf{x}_2$ belongs to S, thereby making $f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2)$ well-defined. Clearly f is concave if and only if $-f$ is convex. When strict inequality holds for every of $\mathbf{x}_1 \neq \mathbf{x}_2$ in S and all t in the open interval $(0,1)$, we use the terms *strictly convex* or *strictly concave* on S. If, for fixed $\mathbf{x}_1$ in S, the preceding inequality holds for every $\mathbf{x}_2$ in S and all $t \in [0,1]$, we say that f is *convex at* $\mathbf{x}_1$. Note that if f is convex on S, then it is convex at each $\mathbf{x}_1$ in S.

In the single-variable setting, we recall using the terms "concave up" and "concave down." In the language of Definition 6.3.2, these merely correspond to "convex" and "concave," respectively.

Convexity has a very simple geometric interpretation in terms of the graph of f. A function is convex on S if and only if given any two distinct inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ in S, the output of f at any convex combination of $\mathbf{x}_1$ and $\mathbf{x}_2$, denoted $\mathbf{x} = t\mathbf{x}_1 + (1-t)\mathbf{x}_2$, where $0 \leq t \leq 1$, must fall on or below the line segment, or chord, connecting the the points $(\mathbf{x}_1, f(\mathbf{x}_1))$ and $(\mathbf{x}_2, f(\mathbf{x}_2))$. (For strict convexity "on or below" is replaced by "strictly below.") Figure 6.5 illustrates this geometric interpretation for the paraboloid function $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by $f(x_1, x_2) = x_1^2 + x_2^2$.

Each of the following functions is convex on its domain:

1. The absolute value function, $f(x) = |x|$.
2. The function $f(x) = |x|^{\frac{3}{2}}$.
3. More generally, the function $f(x) = |x|^p$, where $p \geq 1$.

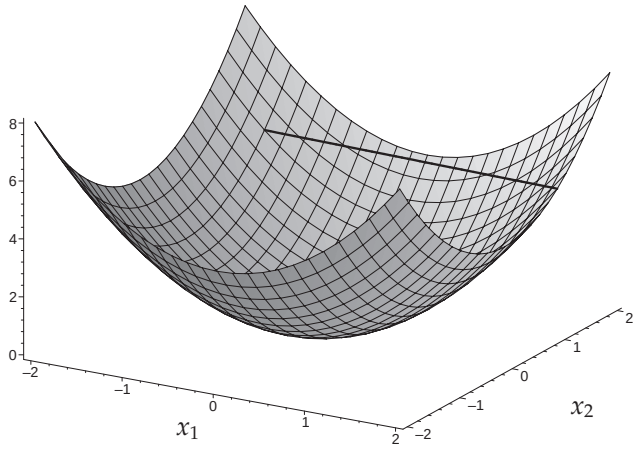


FIGURE 6.5: The paraboloid $f(x_1, x_2) = x_1^2 + x_2^2$, together with one chord illustrating notion of convexity.

4. Any quadratic function, $f(x) = ax^2 + bx + c$, where a, b, and c are real numbers and $a \geq 0$.

5. The Euclidean norm function, $f : \mathbb{R}^n \to \mathbb{R}$ defined as $f(\mathbf{x}) = \|\mathbf{x}\|$.

6. Any *affine transformation*, $f : \mathbb{R}^n \to \mathbb{R}$, defined as $f(\mathbf{x}) = \mathbf{a}^t\mathbf{x} + b$, where $\mathbf{a}$ is a fixed vector in $\mathbb{R}^n$ and b a fixed real number. (This function is both convex and concave.)

6.3.2 Testing for Convexity

The example $f(x) = |x|^{\frac{3}{2}}$ shown in Figure 6.4 illustrates how a function can be differentiable, but not twice-differentiable, on its entire domain and still be convex. From the graphical perspective, we can visualize this convexity not only in terms of the preceding discussion involving chords but also by noting that the tangent line to f at any input, which requires only the first derivative to compute, lies completely below the graph of f. Theorem 6.3.1 uses the gradient and corresponding linearization introduced in Definition 6.2.1 to generalize this result to higher dimensions.

Theorem 6.3.1. Suppose $S \subseteq \mathbb{R}^n$ is convex and that $f : S \to \mathbb{R}$ is differentiable at each point in S. Then f is convex on S if and only if for every $\mathbf{x}_0$ in S,

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) \text{ for all } \mathbf{x} \in S. \tag{6.14}$$

Proof. Assume first that f is convex on S, let $\mathbf{x}_0$ and $\mathbf{x}$ in S, and define $\mathbf{d} = \mathbf{x} - \mathbf{x}_0$. By Exercise 3 from Section 6.2, the differentiability of f implies the existence

of its directional derivatives at $\mathbf{x}_0$. Using the two equivalent formulations of this quantity, we have

$$\begin{aligned}\nabla f(\mathbf{x}_0)^t(\mathbf{x}-\mathbf{x}_0) &= \nabla f(\mathbf{x}_0)^t\mathbf{d}\\ &= \lim_{h\to 0^+}\frac{f(\mathbf{x}_0+h\mathbf{d})-f(\mathbf{x}_0)}{h}\\ &= \lim_{h\to 0^+}\frac{f(h(\mathbf{x}_0+\mathbf{d})+(1-h)\mathbf{x}_0)-f(\mathbf{x}_0)}{h}. \qquad (6.15)\end{aligned}$$

Since f is convex,

$$\begin{aligned}f(h(\mathbf{x}_0+\mathbf{d})+(1-h)\mathbf{x}_0)-f(\mathbf{x}_0) &\le hf(\mathbf{x}_0+\mathbf{d})+(1-h)f(\mathbf{x}_0)-f(\mathbf{x}_0)\\ &= hf(\mathbf{x}_0+\mathbf{d})-hf(\mathbf{x}_0)\\ &= h\left(f(\mathbf{x})-f(\mathbf{x}_0)\right). \qquad (6.16)\end{aligned}$$

Substituting the result from (6.16) into (6.15) leads to (6.14), which completes the first half of the proof.

For the reverse implication, choose $\mathbf{x}_1$ and $\mathbf{x}_2$ in S and $t\in[0,1]$ and assume (6.14) holds. Since S is a convex set, $\mathbf{x}_0 = t\mathbf{x}_1+(1-t)\mathbf{x}_2$ belongs to S. Now apply inequality (6.14) twice, first with $\mathbf{x}=\mathbf{x}_1$, and then with $\mathbf{x}=\mathbf{x}_2$. The results,

$$\begin{aligned}f(\mathbf{x}_1) &\ge f(\mathbf{x}_0)+\nabla f(\mathbf{x}_0)^t(\mathbf{x}_1-\mathbf{x}_0)\\ f(\mathbf{x}_2) &\ge f(\mathbf{x}_0)+\nabla f(\mathbf{x}_0)^t(\mathbf{x}_2-\mathbf{x}_0),\end{aligned}$$

may then be multiplied by the nonnegative quantities t and $1-t$, respectively, and added together. The resulting inequality simplifies to become

$$tf(\mathbf{x}_1)+(1-t)f(\mathbf{x}_2)\ge f(\mathbf{x}_0)=f(t\mathbf{x}_1+(1-t)\mathbf{x}_2).$$

Since $\mathbf{x}_1$ and $\mathbf{x}_2$ were arbitrary in S and t was arbitrary in $[0,1]$, f is convex on S by Definition 6.3.2. This completes the proof of the theorem. □

The preceding proof is easily modified to show that f is convex at $\mathbf{x}_0$ if and only if

$$f(\mathbf{x})\ge f(\mathbf{x}_0)+\nabla f(\mathbf{x}_0)^t(\mathbf{x}-\mathbf{x}_0) \text{ for all } \mathbf{x}\in S.$$

Furthermore, the theorem remains valid if throughout the hypothesis "convex" is replaced by "strictly convex" provided that in (6.14), the inequality is strict and $\mathbf{x}\neq\mathbf{x}_0$.

Theorem 6.3.1 provides one means of establishing a function is convex on a set S that bypasses using the definition. Namely, choose an arbitrary input $\mathbf{x}_0$ in S. Form the linear approximation of f based at $\mathbf{x}_0$:

$$T(\mathbf{x})=f(\mathbf{x}_0)+\nabla f(\mathbf{x}_0)^t(\mathbf{x}-\mathbf{x}_0).$$

Then compute the difference $f(\mathbf{x}) - T(\mathbf{x})$, and establish that this quantity is nonnegative, regardless of the choices of $\mathbf{x}$ and $\mathbf{x}_0$ in S.

For example, if $f(x_1, x_2) = x_1^2 + x_2^2$ and $\mathbf{x}_0 = \begin{bmatrix} x_{0,1} \\ x_{0,2} \end{bmatrix}$, then $\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$ and the linear approximation, or tangent plane, is given by

$$\begin{aligned} T(x_1, x_2) &= (x_{0,1}^2 + x_{0,2}^2) + [2x_{0,1}, 2x_{0,2}] \begin{bmatrix} x_1 - x_{0,1} \\ x_2 - x_{0,2} \end{bmatrix} \\ &= x_{0,1}^2 + x_{0,2}^2 + 2x_{0,1}(x_1 - x_{0,1}) + 2x_{0,2}(x_2 - x_{0,2}) \\ &= -x_{0,1}^2 - x_{0,2}^2 + 2x_1x_{0,1} + 2x_2x_{0,2}. \end{aligned}$$

Thus

$$\begin{aligned} f(x_1, x_2) - T(x_1, x_2) &= x_1^2 + x_2^2 - \left(-x_{0,1}^2 - x_{0,2}^2 + 2x_1x_{0,1} + 2x_2x_{0,2}\right) \\ &= (x_1 - x_{0,1})^2 + (x_2 - x_{0,2})^2. \end{aligned}$$

Since the difference $f - T$ is nonnegative for all choices of $\mathbf{x}$ and $\mathbf{x}_0$ in $\mathbb{R}^2$, we conclude that f is convex on $\mathbb{R}^2$. Figure 6.6 illustrates the nonnegativity of $f - T$ in that the tangent plane to the surface at an arbitrary point lies beneath the surface.

Unfortunately, the preceding example is an elementary one, and Theorem 6.3.1, by itself, is not overly useful for establishing convexity. Fortunately, when f happens to be twice-differentiable on S, a concept we formally define in Section 6.4, there exists a much simpler means for performing this task.

6.3.3 Convexity and *The Global Optimal Solutions Theorem*

Any discussion of convexity is incomplete without citing one of the most important theorems in nonlinear programming, both in the constrained and unconstrained settings. Commonly known as *The Global Optimal Solutions Theorem*, it provides conditions under which a local minimum of an unconstrained NLP is guaranteed to be a global minimum.

Theorem 6.3.2. Assume $S \subseteq \mathbb{R}^n$ is nonempty and convex and that $f : S \to \mathbb{R}$ is convex on S. Suppose $\mathbf{x}_0$ is a local minimum of f on S. Then $\mathbf{x}_0$ is a global minimum of f on S. If f is strictly convex on S or if $\mathbf{x}_0$ is a strict local minimum, then $\mathbf{x}_0$ is a unique global minimum.

Remark: Clearly a similar result holds if throughout this theorem's statement we replace "convex" with "concave," "minimum" with "maximum," and "strictly convex" with "strictly concave."

Proof. Assume first that f has a local minimum at $\mathbf{x}_0$. By Definition (6.2.2),

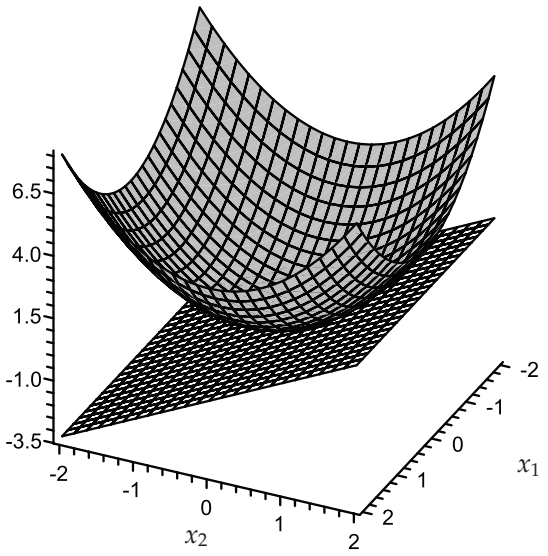


FIGURE 6.6: The paraboloid $f(x_1, x_2) = x_1^2 + x_2^2$, together with the linearization, or tangent plane, at an arbitrary point.

there must exist a sufficiently small positive ϵ such that

$$f(\mathbf{x}_0) \leq f(\mathbf{x}) \text{ for all } \mathbf{x} \text{ in } S \text{ satisfying } \|\mathbf{x} - \mathbf{x}_0\| < \epsilon. \tag{6.17}$$

If $\mathbf{x}_0$ is not a global minimum of f on S, then there must exist in S some $\mathbf{x}_\bigstar$ satisfying $f(\mathbf{x}_\bigstar) < f(\mathbf{x}_0)$. Now consider a convex combination of $\mathbf{x}_0$ and $\mathbf{x}_\bigstar$, denoted $t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0$, where $0 \leq t \leq 1$. Note that this input belongs to S since this set is convex. By the convexity of the function f,

$$\begin{aligned} f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right) &\leq tf(\mathbf{x}_\bigstar) + (1-t)f(\mathbf{x}_0) \\ &< tf(\mathbf{x}_0) + (1-t)f(\mathbf{x}_0) \\ &= f(\mathbf{x}_0). \end{aligned} \tag{6.18}$$

This inequality is valid for all $0 \leq t \leq 1$. Thus, by choosing t_0 sufficiently close to zero, $\tilde{\mathbf{x}} = t_0\mathbf{x}_\bigstar + (1-t_0)\mathbf{x}_0$ satisfies both $f(\tilde{\mathbf{x}}) < f(\mathbf{x}_0)$ and $\|\tilde{\mathbf{x}} - \mathbf{x}_0\| < \epsilon$. But condition (6.17) forces $f(\mathbf{x}_0) \leq f(\tilde{\mathbf{x}})$, and we attain a contradiction. Thus, $\mathbf{x}_0$ must be a global minimum of f.

Now assume that $\mathbf{x}_0$ is a strict local minimum, implying that $f(\mathbf{x}_0) < f(\mathbf{x})$ in (6.17). Then $\mathbf{x}_0$ is a global minimum of f on S by the preceding result. To show it is unique with this property, we assume that $\mathbf{x}_\bigstar$ is a second global

minimum so that $f(\mathbf{x}_\bigstar) = f(\mathbf{x}_0)$. By the convexity of f again, for all $0 \leq t \leq 1$,

$$\begin{aligned} f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right) &\leq tf(\mathbf{x}_\bigstar) + (1-t)f(\mathbf{x}_0) \\ &= f(\mathbf{x}_0). \end{aligned} \tag{6.19}$$

This result is valid for all $0 \leq t \leq 1$. If t_0 is positive and sufficiently close to zero, $\tilde{\mathbf{x}} = t_0\mathbf{x}_\bigstar + (1-t_0)\mathbf{x}_0$ is within ϵ units of $\mathbf{x}_0$, where ϵ is given in (6.17). From (6.17) it follows that $f(\mathbf{x}_0) \leq f(\tilde{\mathbf{x}})$. At the same time, (6.19) dictates $f(\tilde{\mathbf{x}}) \leq f(\mathbf{x}_0)$, from which it follows that these two quantities are equal. Hence, $\mathbf{x}_0$ is not a strict local minimum of f, and therefore $\mathbf{x}_0$ is the unique global minimum of f on S.

The case when f is strictly convex is similar and is as an exercise. This completes the proof. □

6.3.4 Solving the Unconstrained NLP for Differentiable, Convex Functions

Theorem 6.3.2 dictates that for a convex function, the distinction between local and global minima is not necessary. Instead, we may speak simply of global minima and, if the function is strictly convex, the unique global minimum. For differentiable, convex functions, finding these minima is accomplished simply by finding roots of the gradient. Theorem 6.3.3 provides a formal statement of this fact. We omit its proof, which can be found in other sources [2].

Theorem 6.3.3. Assume $S \subseteq \mathbb{R}^n$ is nonempty and convex and that $f : S \to \mathbb{R}$ is convex on S and differentiable everywhere. Then $\mathbf{x}_0$ is the minimum of f on S if and only if $\nabla f(\mathbf{x}_0) = \mathbf{0}$.

With the aid of Theorem 6.3.3, we are on the verge of determining how to solve unconstrained NLPs such as the unconstrained version of the *ConPro Manufacturing Company* NLP given in Section 6.1:

$$\begin{aligned} &\text{maximize} f(x_1, x_2) = 1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} - 350x_1 - 200x_2, \\ &\qquad \text{where } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in S = \mathbb{R}^2_+ = \{(x_1, x_2) \mid x_1, x_2 > 0\}. \end{aligned} \tag{6.20}$$

Recall in this example that x_1 denotes number of units capital, x_2 number of units labor, and f is profit. A surface plot of f is shown in Figure 6.7.

Straightforward calculations establish f is differentiable on S and

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{700x_2^{\frac{1}{3}}}{\sqrt{x_1}} - 350 \\ \dfrac{1400\sqrt{x_1}}{3x_2^{\frac{2}{3}}} - 200 \end{bmatrix}. \tag{6.21}$$

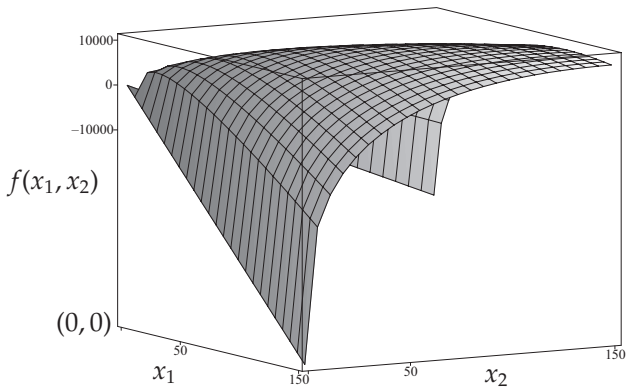


FIGURE 6.7: Surface plot of *ConPro* objective function.

Setting $\nabla f(\mathbf{x}_0) = \mathbf{0}$ and solving for $\mathbf{x}_0$ yields

$$\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix} \approx \begin{bmatrix} 87.11 \\ 101.63 \end{bmatrix}. \tag{6.22}$$

Our objective is to maximize f on S, which is identical to minimizing $-f$. By Theorem 6.3.3, $\mathbf{x}_0$ in (6.22) is the global maximum provided we can establish that $-f$ is convex on S, i.e., f is concave on S. Figure 6.7 suggests that this is the case.

However, using Theorem 6.3.1 to formally verify this fact is extremely challenging, as it requires us to prove that $T - f$ is nonnegative on S, where T is the tangent plane to f at an arbitrary point of S. Fortunately, in Section 6.4 we develop a much simpler means for establishing f is concave, one analogous to the single-variable second derivative test. Before starting that discussion, however, we investigate how convexity arises in the context of regression.

6.3.5 Multiple Linear Regression

Suppose we have a sequence of data pairs,

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m), \tag{6.23}$$

where $\mathbf{x}_i$ belongs to $\mathbb{R}^n$ and y_i belongs to $\mathbb{R}$ for each $1 \leq i \leq m$. If we view each $\mathbf{x}_i$ as an independent variable quantity associated with a dependent variable quantity, y_i, multiple linear regression seeks to find a function, $f : \mathbb{R}^n \to \mathbb{R}$, that best fits this data and that takes the form $f(\mathbf{x}) = \mathbf{a}^t\mathbf{x} + b$, where $\mathbf{a}$ belongs to $\mathbb{R}^n$ and b belongs to $\mathbb{R}$. To determine the function f that best fits the data in the sense of least squares, we must find $\mathbf{a}$ and b that minimize the *sum of*

squared-errors,

$$\sum_{i=1}^{m} (f(\mathbf{x}_i) - y_i)^2 = \sum_{i=1}^{m} \left(\mathbf{a}^t\mathbf{x}_i + b - y_i\right)^2. \tag{6.24}$$

Each term, $\left(\mathbf{a}^t\mathbf{x}_i + b - y_i\right)$, where $1 \leq i \leq m$, is an affine transformation of $\mathbf{a}$ and b and, hence, is convex. Results from exercises (2) and (3) of this section demonstrate that the entire sum, (6.24), is then also convex. By Theorem 6.3.3, it follows that (6.24) is minimized by computing the roots of its gradient in $\mathbf{a}$ and b.

When $n = 1$, then $f(\mathbf{x}) = \mathbf{a}^t\mathbf{x}+b$ reduces to the standard regression line. If $n > 1$, then derivation of f is known as multiple linear regression. A simple example of such a problem is illustrated by the data in Table 6.1, which lists several cigarette brands. To each brand, we associate its tar and nicotine amounts, its mass, and the amount of produced carbon monoxide (CO).

TABLE 6.1: Cigarette data

Brand	Tar (mg.)	Nicotine (mg.)	Mass (gm.)	CO (mg.)
Alpine	14.1	0.86	0.9853	13.6
Benson and Hedges	16	1.06	1.0938	16.6
Bull Durham	29.8	2.03	1.165	23.5
Camel Lights	8	0.67	0.928	10.2
Carlton	4.1	0.4	0.9462	5.4
Chesterfield	15	1.04	0.8885	15
Golden Lights	8.8	0.76	1.0267	9
Kent	12.4	0.95	0.9225	12.3
Kool	16.6	1.12	0.9372	16.3
L and M	14.9	1.02	0.8858	15.4
Lark Lights	13.7	1.01	0.9643	13
Marlboro	15.1	0.9	0.9316	14.4
Merit	7.8	0.57	0.9705	10
Multi Filter	11.4	0.78	1.124	10.2
Newport Lights	9	0.74	0.8517	9.5
Now	1	0.13	0.7851	1.5
Old Gold	17	1.26	0.9186	18.5
Pall Mall Light	12.8	1.08	1.0395	12.6
Raleigh	15.8	0.96	0.9573	17.5
Salem Ultra	4.5	0.42	0.9106	4.9
Tareyton	14.5	1.01	1.007	15.9
TRUE	7.3	0.61	0.9806	8.5
Viceroy Rich Light	8.6	0.69	0.9693	10.6
Virginia Slims	15.2	1.02	0.9496	13.9
Winston Lights	12	0.82	1.1184	14.9

In many cases, data such as that in Table 6.1 is given in spreadsheet format. Maple's `ExcelTools` package provides a convenient means for importing Excel spreadsheet data into Maple array structures. Once the package is loaded, a range of Excel cells is imported as an array using a command of the form

```
Import("C:UsersDocumentsCigaretteData.xls", "Sheet1", "A1:A25").
```

Observe that this command indicates the directory of the Excel file, the file's name, the name of the worksheet in the file, and the cell range. Assuming that the entries from (6.1) are located in the first rows 1-25 and columns 1-4 of the first sheet of the file CigaretteData.xls, we can read the entries into four vectors as follows:

```
> restart: with(ExcelTools): with(Statistics): with(VectorCalculus):
  with(LinearAlgebra):
> tar := convert(Import("C:\\Users\\Documents\\CigaretteData.xls",
  "Sheet1", "A1:A25"), Vector):
> nicotine := convert(Import("C:\\Users
  Documents\\CigaretteData.xls", "Sheet1", "B1:B25"), Vector):
> mass := convert(Import("C:\\Users\\Documents\\CigaretteData.xls",
  "Sheet1", "C1:C25"), Vector):
> CO := convert(Import("C:\\Users\\Documents\\CigaretteData.xls",
  "Sheet1", "D1:D25"), Vector):
```

Here we have converted the arrays to vectors through use of the `convert` command. Doing so allows us to access entries in each vector using a single index. Now we can use this data to determine the linear function of best fit as follows:

```
> x := Matrix(3, 1, [x1, x2, x3]): # Independent variable vector.
> a := Matrix(3, 1, [a1, a2, a3]): # Unknown regression coefficients.
> f:=unapply((Transpose(a).x)[1,1]+b,[x1,x2,x3]):
  # Define linear regression function.
> S:=add((f(tar[i],nicotine[i],mass[i])-CO[i])^2,i=1..25):
  # Sum of squared-errors.
> DelS:=Gradient(S,[a1,a2,a3,b]): # Gradient of S.
> fsolve({DelS[1]=0,DelS[2]=0,DelS[3]=0,DelS[4]=0},{a1,a2,a3,b});
  # Determine critical point of S
```

$$a1 = .96257, a2 = -2.63167, a3 = -.13048, b = 3.20221$$

Thus the linear function that best predicts carbon monoxide output as a function of tar, nicotine, and cigarette mass, based upon the information in Table 6.1, is given by

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{a}^t\mathbf{x} + b \\ &= .96257x_1 - 2.63167x_2 - .13048x_3 + 3.20221 \end{aligned}$$

If we append to the preceding worksheet the appropriate commands, we can determine the *coefficient of determination*, or "R^2 value." This quantity lies between 0 and 1 and measures the proportion of variation of the data accounted for by the function f. If the data pairs used for the regression are given by $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$ and if $\overline{y}$ denotes the mean of $\{y_1, y_2, \ldots, y_m\}$, this proportion equals

$$\frac{\sum_{k=1}^{m} \left(f(\mathbf{x}_k) - y_k\right)^2}{\sum_{k=1}^{m} \left(\overline{y} - y_k\right)^2}. \tag{6.25}$$

The following commands demonstrate a strong goodness-of-fit:

```
> f:=(x1,x2,x3)->.96257*x1-2.63167*x2-.13048*x3+3.20221:# Enter
  best-fit function.
> CObar:=Mean([seq(CO[i],i=1..25)]); # Compute mean carbon monoxide
  value.
```

$$CObar := 12.528$$

```
> Rsquared:=add((f(tar[i],nicotine[i],mass[i])-CObar)^,i=1..25)/
  add((CObar-CO[i])^,i=1..25);
```

$$.91859$$

Exercises Section 6.3

1. Show that if S_1 and S_2 are each convex sets in $\mathbb{R}^n$, then $S_1 \cap S_2$ is also convex.

2. Use Definition 6.3.2 to show that each of the following functions is convex on its given domain.

   (a) The absolute value function, $f(x) = |x|$. (Hint: Use the triangle inequality.)

   (b) Any quadratic function, $f(x) = ax^2 + bx + c$, where a, b, and c are real numbers and $a \geq 0$.

   (c) The Euclidean norm function, $f : \mathbb{R}^n \to \mathbb{R}$ defined as $f(\mathbf{x}) = \|\mathbf{x}\|$.

3. Show that the *affine transformation*, $f : \mathbb{R}^n \to \mathbb{R}$, defined as $f(\mathbf{x}) = \mathbf{a}^t\mathbf{x} + b$, for some fixed vector, $\mathbf{a}$, in $\mathbb{R}^n$ and fixed real number, b, is both convex and concave on $\mathbb{R}^n$.

4. Show that $f(x) = x^4$ is strictly convex on $\mathbb{R}$. (Hint: Use Theorem 6.3.1.)

5. Suppose that $f(x_1, x_2) = 2x_1^2x_2 + x_1^2x_2^2 - 4x_1x_2 - 2x_1x_2^2$. Show that f is neither convex nor concave on $\mathbb{R}^2$. (Hint: First fix x_1, say $x_1 = 1$, and consider the graph of $f(1, x_2)$. Then fix $x_2 = 1$ and consider the graph of $f(x_1, 1)$.)

6. Suppose $f : S \to \mathbb{R}$ and $g : S \to \mathbb{R}$ are both convex. Show that $\alpha f + \beta g$ is then convex on S for any nonnegative values of α and β.

7. Suppose $S \subseteq \mathbb{R}^n$ and that $f : S \to \mathbb{R}$ is a convex function. If $\psi : \mathbb{R} \to \mathbb{R}$ is convex on $f(S)$, show that the composite, $\psi \circ f$, is convex on S.

8. Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable on $\mathbb{R}^n$, A is an n-by-n matrix, and $\mathbf{b}$ a vector in $\mathbb{R}^n$. Define $\phi : \mathbb{R}^n \to \mathbb{R}$ by $\phi(\mathbf{x}) = f(A\mathbf{x} + \mathbf{b})$. Show that ϕ is convex on $\mathbb{R}^n$ and determine a formula for $\nabla\phi(\mathbf{x})$ in terms of ∇f.

9. Suppose that $S \subseteq \mathbb{R}^n$ is convex and that $f : S \to \mathbb{R}$ is a convex function. For fixed $y \in \mathbb{R}$, we define the corresponding *lower-level set* as

$$f^{-1}(y) = \{\mathbf{x} \mid \mathbf{x} \in S \text{ and } f(\mathbf{x}) \leq y\}. \tag{6.26}$$

   Prove that $f^{-1}(y)$ is a convex set for every $y \in \mathbb{R}$.

10. Suppose that the constraint functions $g_1, g_2, \ldots, g_m$ in the general NLP (6.1) are all convex. Show that the feasible region of the NLP is then a convex set. (Hint: Combine the result from the previous exercise with that of Exercise 1.)

11. The *Body Fat Index* (BFI) is one means of measuring an individual's fitness. One method of computing this value is Brozek's formula, which defines

$$\text{BFI} = \frac{457}{\rho} - 414.2,$$

    where ρ denotes the body density in units of kilograms per liter. Unfortunately, an accurate measurement of ρ can only be accomplished through a process known as hydrostatic weighing, which requires recording an individual's weight while under water, with all air expelled from his or her lungs. In an effort to devise less time-consuming means for estimating BFI, researchers have collected data that suggests a linear relationship between BFI and various body measurements [38]. Table 6.2 lists the weight, height, and various body measurements for ten individuals. The last column indicates the BFI for these same individuals, as determined by the process of hydrostatic weighing and Brozek's formula.

    Let $\mathbf{x}$ denote a vector in $\mathbb{R}^5$ that records a person's weight, height, abdomen circumference, wrist circumference, and neck circumference.

Use multiple linear regression to find the linear function $f : \mathbb{R}^n \to \mathbb{R}$, given by $f(\mathbf{x}) = \mathbf{a}^t\mathbf{x}$, which best fits the given data, in the sense of least-squares, and can be used to predict an individual's body fat index, based upon the five obtained measurements types. Calculate the corresponding coefficient of determination.

TABLE 6.2: Body measurement data

Weight (lb.)	Height (in.)	Abdomen (cm.)	Wrist (cm.)	Neck (cm.)	BFI
154.25	67.75	85.2	17.1	36.2	12.6
173.25	72.25	83	18.2	38.5	6.9
154	66.25	87.9	16.6	34	24.6
184.75	72.25	86.4	18.2	37.4	10.9
184.25	71.25	100	17.7	34.4	27.8
210.25	74.75	94.4	18.8	39	20.6
181	69.75	90.7	17.7	36.4	19
176	72.5	88.5	18.8	37.8	12.8
191	74	82.5	18.2	38.1	5.1
198.25	73.5	88.6	19.2	42.1	12

6.4 Sufficient Conditions for Local and Global Optimal Solutions

The second derivative test in calculus is based upon the following argument. If a function f is twice-differentiable at x_0, then the quadratic Taylor polynomial approximation of f at x_0 is given by

$$P_2(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2. \tag{6.27}$$

When $f'(x_0) = 0$,

$$f(x) \approx f(x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 \text{ for } x \approx x_0. \tag{6.28}$$

The sign of $f''(x_0)$ indicates whether locally, near x_0, the graph of f is parabolic opening up, indicating x_0 is a local minimum, or parabolic opening down, indicating x_0 is a local maximum. Of course, these two cases correspond to the graph of f being convex or concave, respectively at x_0. If $f''(x_0) = 0$, no conclusion may be made regarding the nature of the critical point; other means of investigation are required.

6.4.1 Quadratic Forms

Since quadratic functions hold the key to the second derivative test, we investigate their higher-dimensional analog. In $\mathbb{R}^n$, this type of function is known as a *quadratic form*. Before defining this term we recall that a symmetric matrix, A, is one satisfying $A^t = A$.

Definition 6.4.1. A quadratic form on $\mathbb{R}^n$ is a function $f : \mathbb{R}^n \to \mathbb{R}$ represented as

$$f(\mathbf{x}) = \mathbf{x}^t A \mathbf{x},$$

for some n-by-n symmetric matrix A.

We usually refer to A as the *matrix* associated with the quadratic form. When $n = 1$, this quantity is merely the coefficient a of the quadratic power function $f(x) = ax^2$.

It is because the matrix, A, associated with the quadratic form is symmetric that the gradient of f takes a particularly simple form. Namely, $\nabla f(\mathbf{x}) = 2A\mathbf{x}$. We will use this fact frequently throughout subsequent discussions.

Figures 6.8-6.10 illustrate three different quadratic forms having domain $\mathbb{R}^2$. Observe how their shapes differ.

1.

$$\begin{aligned} f(x_1, x_2) &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= 2x_1^2 + 2x_1x_2 + 3x_2^2 \end{aligned} \tag{6.29}$$

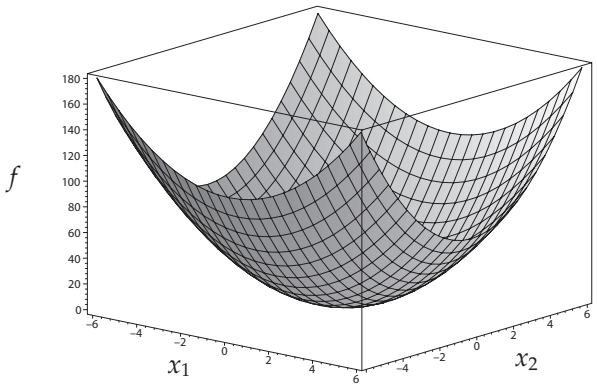


FIGURE 6.8: The quadratic form $f(x_1, x_2) = 2x_1^2 + 2x_1x_2 + 3x_2^2$.

2.

$$\begin{aligned} f(x_1, x_2) &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -2 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= -2x_1^2 + 2x_1x_2 - 3x_2^2 \end{aligned} \tag{6.30}$$

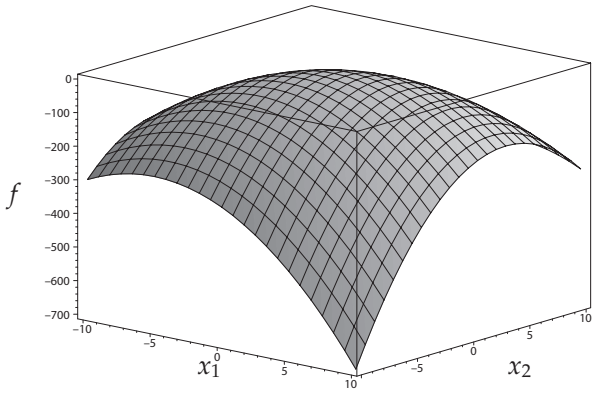


FIGURE 6.9: Quadratic form $f(x_1, x_2) = -2x_1^2 + 2x_1x_2 - 3x_2^2$.

3.

$$\begin{aligned} f(x_1, x_2) &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= x_1^2 + 4x_1x_2 + x_2^2 \end{aligned} \tag{6.31}$$

6.4.2 Positive Definite Quadratic Forms

The examples shown in Figures 6.8-6.10 illustrate the three fundamental types of quadratic forms having domain $\mathbb{R}^2$: paraboloids opening up, paraboloids opening down, and saddle-shaped surfaces. This trichotomy extends to higher dimensions and is described in the following manner.

Definition 6.4.2. A quadratic form f on $\mathbb{R}^n$ is *positive definite* if and only if

$$f(\mathbf{x}) > 0 \text{ for every } \mathbf{x} \neq \mathbf{0}.$$

The quadratic form is *negative definite* if and only if

$$f(\mathbf{x}) < 0 \text{ for every } \mathbf{x} \neq \mathbf{0} \text{ or, equivalently, if } -f \text{ is positive definite.}$$

Finally, f is *indefinite* if $f(\mathbf{x})$ is positive for some choices $\mathbf{x}$ and negative for other choices.

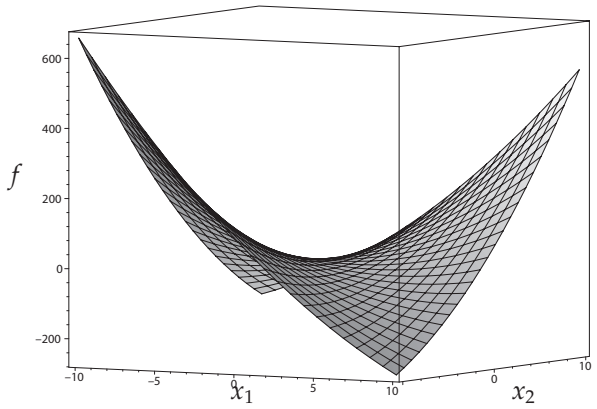


FIGURE 6.10: Quadratic form $f(x_1, x_2) = x_1^2 + 4x_1x_2 + x_2^2$.

Establishing whether a given quadratic form is positive definite, negative definite, or indefinite using only Definition 6.4.2 can prove quite challenging. Fortunately, an alternate means exists, one that capitalizes upon the fact that every quadratic form is determined completely by a square symmetric matrix. The following theorem, whose proof we omit but which can be found in a variety of sources, uses eigenvalues as a tool for classifying quadratic forms [15], [6]. Recall that an eigenvalue of an n-by-n matrix A is a scalar, λ, satisfying $A\mathbf{x} = \lambda\mathbf{x}$ for some nonzero vector $\mathbf{x} \in \mathbb{R}^n$. For a symmetric matrix, all eigenvalues are real-valued. (See Appendix **??**.)

Theorem 6.4.1. Suppose A is a symmetric n by n matrix having real-valued entries and that
$f : \mathbb{R}^n \to \mathbb{R}$ is the quadratic form given by

$$f(\mathbf{x}) = \mathbf{x}^t A\mathbf{x}.$$

Then f is positive definite (resp. negative definite) if and only if all eigenvalues of A are positive (resp. negative).

◆ ◆

Waypoint 6.4.1. Determine the eigenvalues of the matrices associated with the quadratic forms in Figures 6.8 and 6.9. Use your results to classify each as positive definite or negative definite.

◆ ◆

Theorem 6.4.1 is not complete in terms of classifying all quadratic forms. A slight modification of Definition 6.4.2 avoids this problem. Namely, we relax

the condition of strict inequality and define the quadratic form f to be *positive semidefinite* if and only if

$$f(\mathbf{x}) \geq 0 \text{ for every } \mathbf{x} \neq \mathbf{0}.$$

A modification of the statement of the preceding theorem classifies a quadratic form as positive semidefinite if all eigenvalues are nonnegative. The notion of *negative semidefinite* is defined analogously. Finally we say that the quadratic form of f is *indefinite* if it is neither positive nor negative semidefinite, meaning it has both positive and negative eigenvalues.

With one more additional tool, we shall discover how these three types of quadratic forms hold the key for constructing our higher-dimensional version of the second derivative test.

6.4.3 Second-order Differentiability and the Hessian Matrix

In Section 6.2, we defined first-order differentiability, which led to the concept of linear approximations. Second-order differentiability is defined similarly. However, while the gradient vector played the role of the first derivative, the role of the second derivative is played by a square matrix.

Definition 6.4.3. Suppose that $S \subseteq \mathbb{R}^n$ and that $f : S \to \mathbb{R}$ is differentiable at $\mathbf{x}_0$, which lies in the interior of S. We say that the function f is *twice-differentiable* at $\mathbf{x}_0$, or is *second-order differentiable* at $\mathbf{x}_0$, if and only if there exists an n-by-n matrix, $H_f(\mathbf{x}_0)$, called the *Hessian matrix of f at $\mathbf{x}_0$*, and a scalar function R, depending upon $\mathbf{x}_0$ and $\mathbf{x}$, such that

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^t H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \|\mathbf{x} - \mathbf{x}_0\|^2 R(\mathbf{x}_0; \mathbf{x}), \quad (6.32)$$

where $\lim_{\mathbf{x} \to \mathbf{x}_0} R(\mathbf{x}_0; \mathbf{x}) = 0$.

In the same way that first-order differentiability forces the gradient vector to be comprised of first-order partial derivatives, second-order differentiability leads to a Hessian matrix consisting of second-order partial derivatives:

$$H_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}. \quad (6.33)$$

Whenever referring to the Hessian, H_f, we will assume that f has continuous

second-order partial derivatives on the set S, i.e., is f is *continuously twice-differentiable* there. In this case, *Clairaut's Theorem* implies equality of mixed second-order partial derivatives: $\frac{\partial^2 f}{\partial x_i x_j} = \frac{\partial^2 f}{\partial x_j x_i}$ for all $1 \leq i, j \leq n$, so that $H_f(\mathbf{x})$ is a symmetric matrix and the right-hand side of (6.32) therefore contains a quadratic form as one of its terms. Naturally we refer to the right-hand without the remainder term, as the second-order approximation of f near $\mathbf{x}_0$. It is analogous to the degree-two Taylor polynomial from the single-variable setting. Finally, we note that Definitions 6.4.1 and 6.4.3 together imply that the Hessian matrix of a quadratic form $f(\mathbf{x}) = \mathbf{x}^t A\mathbf{x}$ equals $2A$.

In Maple, the `Hessian` command, located in the `VectorCalculus` package provides a means for computing the Hessian matrix. Its general form is given by `Hessian(`*`expression`*`,`*`variable list`*`)`. Here, *`variable list`* indicates the variables with respect to which the Hessian of *`expression`* is computed. The output of the command is a matrix. A simple example of the command's usage is as follows:

```
> with(VectorCalculus):
> Hessian(x1^2 +x1*x2^3,[x1,x2]);
```

$$\begin{bmatrix} 2 & 3x_2^2 \\ 3x_2^2 & 6x_1x_2 \end{bmatrix}$$

If f is a function, the `unapply` command can be used to define the corresponding Hessian function. For example, if $f(x_1, x_2) = x_1^2 + x_2^2$, its Hessian matrix function is constructed using the following syntax. Here we name this function `Hf`.

```
> with(VectorCalculus):
> f:=(x1,x2)->x1^2 +x1*x2^3,[x1,x2];
```

$$f := (x_1, x_2) \rightarrow x_1^2 + x_1x_2^3 :$$

```
> Hf:=unapply(Hessian(f(x1,x2),[x1,x2]),[x1,x2]):
> Hf(x1,x2);
```

$$\begin{bmatrix} 2 & 3x_2^2 \\ 3x_2^2 & 6x_1x_2 \end{bmatrix}$$

Waypoint 6.4.2. For each of the following functions, determine a general formula for the Hessian matrix, $H_f(\mathbf{x})$. Then, determine the critical point, $\mathbf{x}_0$, of each function, and evaluate $H_f(\mathbf{x}_0)$.

1. The function $f(x_1, x_2) = x_1x_2^2 - 2x_1^2 + 3x_2$.
2. The objective function associated with the *ConPro Manufacturing Company*. (Note: Results (6.21) and (6.22) from Section 6.3. may prove useful.)

The Hessian is an extremely useful tool for classifying local optimal solutions and also for establishing a function is convex on its domain. Theorem 6.4.2, our long-desired "second derivative test," addresses the first of these issues.

Before stating this theorem, we describe an important result from linear algebra that we will utilize in its proof.

Definition 6.4.4. A square, invertible matrix, P, satisfying $P^{-1} = P^t$ is said to be an *orthogonal matrix*.

An orthogonal n-by-n matrix, P, has interesting properties. Among them are the following:

- The column vectors of P are orthonormal, meaning that $\mathbf{u}_i^t\mathbf{u}_j$ equals zero for $1 \leq i, j \leq n$ such $i \neq j$ and equals 1 if $i = j$.
- $||P\mathbf{x}|| = ||\mathbf{x}||$ for any $\mathbf{x}$ in $\mathbb{R}^n$.
- The matrix P^t is orthogonal as well.

Orthogonal matrices play an important role in matrix diagonalization. The Spectral Theorem (Theorem B.7.1) states that any symmetric matrix, A, can be expressed in the form

$$A = PDP^t, \tag{6.34}$$

where D is a diagonal matrix whose entries are the eigenvalues of A and P is an orthogonal matrix whose columns consist of the eigenvectors of A. Further discussion of orthogonal matrices can be found in a variety of sources [21], [15].

We now state our main result.

Theorem 6.4.2. Assume $S \subseteq \mathbb{R}^n$ is a nonempty, open, convex set. Suppose $f : S \to \mathbb{R}$ is twice-differentiable at each point of S and that $\mathbf{x}_0$ is a critical point of f belonging to S.

1. If f has a local minimum (resp. local maximum) at $\mathbf{x}_0$, then $H_f(\mathbf{x}_0)$ is positive semidefinite (resp. negative semidefinite).
2. If $H_f(\mathbf{x}_0)$ is positive definite (resp. negative definite), then $\mathbf{x}_0$ is a strict local minimum (resp. strict local maximum).
3. If $H_f(\mathbf{x}_0)$ is indefinite, then $\mathbf{x}_0$ is neither a maximum nor minimum. It is termed a *saddle point*.
4. If $H_f(\mathbf{x}_0)$ is positive semidefinite or negative semidefinite, then no conclusion may be made as to the nature of the critical point $\mathbf{x}_0$. Further analysis is required.

Proof. We will prove (1), (2), and (3); (4) will be left as an exercise.

Assume first that f has a local minimum at $\mathbf{x}_0$. Since f is twice-differentiable and has a $\mathbf{x}_0$ is a critical point, we have for all $\mathbf{x}$ in S,

$$\begin{aligned} 0 &\leq f(\mathbf{x}) - f(\mathbf{x}_0) \\ &= \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^t H_f(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) + \|\mathbf{x}-\mathbf{x}_0\|^2 R(\mathbf{x}_0;\mathbf{x}), \end{aligned} \tag{6.35}$$

where $R(\mathbf{x}_0;\mathbf{x}) \to 0$ as $\mathbf{x} \to \mathbf{x}_0$.

Choose $\mathbf{d}$ arbitrary in $\mathbb{R}^n$. Then for sufficiently small, positive h, $\mathbf{x} = \mathbf{x}_0 + h\mathbf{d}$ belongs to S. Substituting this value into (6.35), simplifying the result, and dividing by h^2, we arrive at

$$\begin{aligned} 0 &\leq \frac{f(\mathbf{x}) - f(\mathbf{x}_0)}{h^2} \\ &= \frac{1}{2}\mathbf{d}^t H_f(\mathbf{x}_0)\mathbf{d} + \|\mathbf{d}\|^2 R(\mathbf{x}_0;\mathbf{x}_0 + h\mathbf{d}). \end{aligned} \tag{6.36}$$

This result implies that

$$\frac{1}{2}\mathbf{d}^t H_f(\mathbf{x}_0)\mathbf{d} + \|\mathbf{d}\|^2 R(\mathbf{x}_0;\mathbf{x}_0 + h\mathbf{d}) \geq 0 \tag{6.37}$$

for all sufficiently small, positive h. Letting $h \to 0$, we arrive at $\mathbf{d}^t H_f(\mathbf{x}_0)\mathbf{d} \geq 0$. Since $\mathbf{d}$ was an arbitrary vector in $\mathbb{R}^n$, $H_f(\mathbf{x}_0)$ is positive semidefinite by definition. This completes the proof of (1).

Now assume that $H_f(\mathbf{x}_0)$ is positive definite. Since f is twice-differentiable at $\mathbf{x}_0$ and $\nabla f(\mathbf{x}_0) = \mathbf{0}$, we have

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^t H_f(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) + \|\mathbf{x}-\mathbf{x}_0\|^2 R(\mathbf{x}_0;\mathbf{x}), \tag{6.38}$$

where $\lim_{\mathbf{x}\to\mathbf{x}_0} R(\mathbf{x}_0;\mathbf{x}) = 0$. By (6.34), $H_f(\mathbf{x}_0)$ factors as

$$H_f(\mathbf{x}_0) = PDP^t. \tag{6.39}$$

In this factorization D is a diagonal matrix having the eigenvalues of $H_f(\mathbf{x}_0)$ as its entries, and P has column vectors equal to the corresponding orthonormal eigenvectors of $H_f(\mathbf{x}_0)$. Since $H_f(\mathbf{x}_0)$ is positive definite, all its eigenvalues are strictly positive. Listing the eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_n$, we define $\lambda = \min\{\lambda_i \mid 1 \leq i \leq n\}$ and note that $\lambda > 0$.

We now prove

$$(\mathbf{x} - \mathbf{x}_0)^t H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \geq \lambda \|\mathbf{x} - \mathbf{x}_0\|^2, \tag{6.40}$$

a result, when combined with (6.38), leads to

$$f(\mathbf{x}) - f(\mathbf{x}_0) \geq \|\mathbf{x} - \mathbf{x}_0\|^2 \left(\frac{\lambda}{2} + R(\mathbf{x}_0; \mathbf{x})\right).$$

Since $\lambda > 0$ and $R(\mathbf{x}_0; \mathbf{x}) \to 0$ as $\mathbf{x} \to \mathbf{x}_0$, we will then be able to conclude that $f(\mathbf{x}) - f(\mathbf{x}_0) > 0$ for all $\mathbf{x}$ sufficiently close to $\mathbf{x}_0$, which is precisely what it means to say that $\mathbf{x}_0$ is a strict local minimum of f.

To establish (6.40), we first define

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = P^t(\mathbf{x} - \mathbf{x}_0). \tag{6.41}$$

Since P^t is also orthogonal, $\|\mathbf{w}\| = \|\mathbf{x} - \mathbf{x}_0\|$. We now use this fact to obtain the desired lower bound in (6.40):

$$\begin{aligned}
(\mathbf{x} - \mathbf{x}_0)^t H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) &= (\mathbf{x} - \mathbf{x}_0)^t P D P^t (\mathbf{x} - \mathbf{x}_0) \\
&= \left(P^t(\mathbf{x} - \mathbf{x}_0)\right)^t D P^t (\mathbf{x} - \mathbf{x}_0) \\
&= \mathbf{w}^t D \mathbf{w} \\
&= \lambda_1 w_1^2 + \lambda_2 w_2^2 + \ldots + \lambda_n w_n^2 \\
&\geq \lambda \left(w_1^2 + w_2^2 + \ldots + w_n^2\right) \\
&= \lambda \|\mathbf{w}\|^2 \\
&= \lambda \|\mathbf{x} - \mathbf{x}_0\|^2.
\end{aligned}$$

Thus, we have established (6.40) so that $\mathbf{x}_0$ is a local minimum and the proof of (2) is complete.

If $H_f(\mathbf{x}_0)$ is indefinite, then there exist $\mathbf{x}_+$ and $\mathbf{x}_-$ in S such that $\mathbf{x}_+^t H_f(\mathbf{x}_0)\mathbf{x}_+ > 0$ and $\mathbf{x}_-^t H_f(\mathbf{x}_0)\mathbf{x}_- < 0$. Define $\phi(h) = f(\mathbf{x}_0 + h\mathbf{x}_+)$. By the result from Exercise 4 of Section 6.2, along with a simple extension of this result to the second derivative, ϕ is twice-differentiable in a neighborhood of the origin where it satisfies

$$\phi'(h) = \nabla f(\mathbf{x}_0 + h\mathbf{x}_+)^t \mathbf{x}_+ \text{ and } \phi''(h) = \mathbf{x}_+^t H_f(\mathbf{x}_0 + h\mathbf{x}_+)\mathbf{x}_+.$$

Since $\phi'(0) = 0$ and $\phi''(0) > 0$, ϕ has a local minimum of $\phi(0) = f(\mathbf{x}_0)$ at $h = 0$, from which it follows that f cannot have a local maximum at $\mathbf{x}_0$. A similar argument using $\mathbf{x}_-$ establishes that f cannot have a local minimum at $\mathbf{x}_0$. This completes the proof of (3). □

◆ ——— ◆

Waypoint 6.4.3. Use the Theorem 6.4.2 to classify, where possible, all strict local maxima and minima of the functions corresponding Figures 6.8-6.9.

◆ ——— ◆

While Theorem 6.4.2 is useful for determining local maxima and minima, it cannot be used by itself to determine global optimal solutions of the unconstrained NLP in which we seek to minimize $f(\mathbf{x})$, for $\mathbf{x}$ belonging to some open, convex set S in $\mathbb{R}^n$. The difficulty lies in making the transition from "local" to "global." Fortunately, the Global Optimal Solutions Theorem (Theorem 6.3.2) helps us make this transition, provided we show f is convex on S. This task appeared quite difficult in general at the end of Section 6.3. The next result demonstrates that a more straightforward method exists, one that requires showing $H_f(\mathbf{x}_0)$ is positive semidefinite for all $\mathbf{x}_0$ in S, not just at the critical point.

Theorem 6.4.3. Assume $S \subseteq \mathbb{R}^n$ is nonempty, open, convex set. Suppose $f : S \to \mathbb{R}$ is twice-differentiable at each point of S. Then f is convex on S if and only if $H_f(\mathbf{x}_0)$ is positive semidefinite for every $\mathbf{x}_0$ in S.

Proof. First assume f is convex on S and choose $\mathbf{x}_0$ in this set. To establish $H_f(\mathbf{x}_0)$ is positive semidefinite, we must show that for arbitrary $\mathbf{x}$ in S, $\mathbf{x}^t H_f(\mathbf{x}_0)\mathbf{x} \geq 0$.

Since S is open, for all h sufficiently close to 0, $\mathbf{x}_0 + h\mathbf{x}$ also belongs to S. Thus, by Theorem 6.3.1,

$$f(\mathbf{x}_0 + h\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t (h\mathbf{x}). \tag{6.42}$$

That f is twice-differentiable at $\mathbf{x}_0$ yields

$$f(\mathbf{x}_0 + h\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t (h\mathbf{x}) + \frac{1}{2}(h\mathbf{x})^t H_f(\mathbf{x}_0)(h\mathbf{x}) + \|h\mathbf{x}\|^2 R\left(\mathbf{x}_0; \mathbf{x}_0 + h\mathbf{x}\right). \tag{6.43}$$

The combination of (6.42) and (6.43) leads to

$$\frac{1}{2}h^2 \mathbf{x}^t H_f(\mathbf{x}_0)(\mathbf{x}) + h^2 \|\mathbf{x}\|^2 R\left(\mathbf{x}_0; \mathbf{x}_0 + h\mathbf{x}\right) \geq 0,$$

which is valid for all sufficiently small positive h. Dividing by h^2, letting

$h \to 0$, and utilizing the fact $R(\mathbf{x}; \mathbf{x}_0 + h\mathbf{x}_0) \to 0$, yields the desired inequality $\mathbf{x}^t H_f(\mathbf{x}_0)\mathbf{x} \geq 0$. This completes the first half of the proof.

For the reverse implication, we assume $H_f(\mathbf{x}_0)$ is positive semidefinite and must establish that f is convex on S. Choose $\mathbf{x}_0$ and $\mathbf{x}$ in S and define $\phi : [0, 1] \to \mathbb{R}$ by

$$\phi(h) = f\left((1-h)\mathbf{x}_0 + h\mathbf{x}\right).$$

Note that $\phi(0) = f(\mathbf{x}_0)$ and $\phi(1) = f(\mathbf{x})$. Since f is twice-differentiable on the open set S, ϕ is twice-differentiable in an open set containing the entire closed interval $[0, 1]$. By Taylor's Theorem,

$$\phi(1) = \phi(0) + \phi'(0) + \frac{\phi''(\zeta)}{2} \tag{6.44}$$

for some ζ in $[0, 1]$. Again using the result of Exercise 4 from 6.2, we have $\phi'(0) = \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0)$. An analogous identity for ϕ'', which we leave as an exercise, is given by

$$\phi''(\zeta) = (\mathbf{x} - \mathbf{x}_0)^t H_f\left((1-\zeta)\mathbf{x}_0 + \zeta\mathbf{x}\right)(\mathbf{x} - \mathbf{x}_0). \tag{6.45}$$

Substituting this formula into (6.44) results in

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^t H_f\left((1-\zeta)\mathbf{x}_0 + \zeta\mathbf{x}\right)(\mathbf{x} - \mathbf{x}_0).$$

Because $H_f\left((1-\zeta)\mathbf{x}_0 + \zeta\mathbf{x}\right)$ is positive semidefinite, it follows that

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0).$$

By Theorem 6.3.1, f is convex. □

Theorem 6.4.3 and our previous results provides a framework for solving unconstrained NLPs. Suppose $S \subseteq \mathbb{R}^n$ is an open convex set and that $f : S \to \mathbb{R}$ is twice-differentiable at each point of S. Consider the general unconstrained NLP

$$\text{minimize } f(\mathbf{x}), \text{ where } \mathbf{x} \in S. \tag{6.46}$$

The process of determining the global minimum is as follows:

1. Determine all critical points of f in S, that is, all solutions, $\mathbf{x}_0$, of $\nabla f(\mathbf{x}_0) = \mathbf{0}$. Theorem 6.2.1 dictates that this condition must hold at the optimal solution.

2. Evaluate the Hessian matrix $H_f(\mathbf{x})$ at each of the critical points from (1) and determine which resulting matrices are positive definite. (Computing eigenvalues is one method for accomplishing this.) Then apply Theorem 6.4.2 to determine all local minima.

3. Now ascertain whether f is convex on all of S. If this holds, each local minimum from (2) is a global minimum by the Global Optimal Solutions Theorem (Theorem 6.3.2). To establish convexity of f on S, it suffices to take the general formula for $H_f(\mathbf{x})$ from the previous step and prove that it is positive semidefinite for all $\mathbf{x}$ in S. One means of doing so is to compute the eigenvalues, which are expressions in $\mathbf{x}$, and establish they are all nonnegative.
4. If in (3), $H_f(\mathbf{x})$ is also positive definite for all $\mathbf{x}$ in S, then f is strictly convex on S, implying there can only be one global minimum by the Global Optimal Solutions Theorem.

The unconstrained *ConPro Manufacturing Company* NLP,

$$\text{maximize } f(x_1, x_2) = 1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} - 350x_1 - 200x_2,$$
$$\text{where } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in S = \mathbb{R}^2_+ = \{(x_1, x_2) \mid x_1, x_2 > 0\}. \quad (6.47)$$

illustrates this process. Here we seek to maximize so "negative semidefinite" replaces "positive semidefinite" and "concave" replaces "convex" in the preceding outline of steps. Recall from Section 6.3, the single critical point of f is given by

$$\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix} \approx \begin{bmatrix} 87.11 \\ 101.63 \end{bmatrix}.$$

The Hessian matrix $H_f(\mathbf{x})$ simplifies to

$$H_f(x_1, x_2) = \begin{bmatrix} \dfrac{-350x_2^{\frac{1}{3}}}{x_1^{\frac{3}{2}}} & \dfrac{700}{3\sqrt{x_1}x_2^{\frac{2}{3}}} \\ \dfrac{700}{3\sqrt{x_1}x_2^{\frac{2}{3}}} & \dfrac{-2800\sqrt{x_1}}{9x_2^{\frac{5}{3}}} \end{bmatrix}. \quad (6.48)$$

Computing the eigenvalues of $H_f(x_1, x_2)$ is a tedious task that is best left to Maple. The resulting eigenvalues are then given by

$$-\frac{175}{9}\frac{9x_2^2x_1 + 8x_1^3 \pm \sqrt{81x_2^4x_1^2 + 64x_1^6}}{x_2^{5/3}x_1^{5/2}}. \quad (6.49)$$

Clearly one of these eigenvalues is negative for all $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ in S. To establish the other is negative on S, we must show that for such $\mathbf{x}$,

$$9x_2^2x_1 + 8x_1^3 - \sqrt{81x_2^4x_1^2 + 64x_1^6} > 0. \quad (6.50)$$

Waypoint 6.4.4. Show that the preceding quantity is negative in S. (Hint: First establish the general inequality $\sqrt{a+b} < a+b$ for $a, b > 0$.)

Since both eigenvalues are negative on S, $H_f(\mathbf{x})$ is negative definite on S so that f is strictly concave. Thus, the unique global minimum is given by Thus

$$\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix} \approx \begin{bmatrix} 87.11 \\ 101.63 \end{bmatrix}.$$

Finally, we note that the Maple command `IsDefinite` provides a shortcut means for determining whether a given square matrix is positive- or negative-definite or semidefinite. Its general form consists of `IsDefinite(M,'query'=q)`, where `M` is a symmetric matrix and `q` specifies the matrix form to be determined. Choices for `q` consist of `'positive_definite'`, `'positive_semidefinite'`, `'negative_definite'`, `'negative_semidefinite'`, and `'indefinite'`. The returned value is *true* or *false*; if the matrix has symbolic entries, the command returns conditions on these entries for which `q` is satisfied.

6.4.4 Using Maple To Classify Critical Points for the Unconstrained NLP

The following worksheet, **Classifying Critical Points.mw**, demonstrates how various Maple commands are combined to determine and classify the critical point of the unconstrained *ConPro Manufacturing Company* NLP.

```
> with(VectorCalculus):with(LinearAlgebra):
> f:=(x1,x2)->1400*x1^(1/2)*x2^(1/3)-350*x1-200*x2;
  # Enter function to maximized.
```

$$f := (x1, x2) \to 1400\sqrt{x_1}x_2^{1/3} - 350x_1 - 200x_2$$

```
> Delf:=unapply(Gradient(f(x1,x2),[x1,x2]),[x1,x2]):
  # Create the gradient function of f.
> solve({Delf(x1,x2)[1]=0,Delf(x1,x2)[2]=0},{x1,x2});
  # Determine the critical point by solving a system of two equations
  in x1 and x2. These are formed by setting each component of the
  gradient of f equal to 0.
```

$$\{x_1 = 784/9, x_2 = 2744/27\}$$

```
> evalf(%);
  # Determine floating point approximation of critical point.
```

$$\{x_1 = 87.1111, x_2 = 101.6296\}$$

```
> subs(%,f(x1,x2));
  # Determine objective value at critical point.
```

$$10162.96296$$

```
> Hf:=Hessian(f(x1,x2),[x1,x2]);
  # Create a matrix, Hf, consisting of the Hessian of f.
```

$$Hf = \begin{bmatrix} -350\frac{\sqrt[3]{x_2}}{x_1^{3/2}} & \frac{700}{3}\frac{1}{\sqrt{x_1}x_2^{2/3}} \\ \frac{700}{3}\frac{1}{\sqrt{x_1}x_2^{2/3}} & -\frac{2800}{9}\frac{\sqrt{x_1}}{x_2^{5/3}} \end{bmatrix}$$

```
> Eigenvalues(Hf);
  # Determine eigenvalues of Hf.
```

$$\begin{bmatrix} -\frac{175}{9}\frac{8x_1^3+9x_1x_2^2-\sqrt{64x_1^6+81x_1^2x_2^4}}{x_2^{5/3}x_1^{5/2}} \\ -\frac{175}{9}\frac{8x_1^3+9x_1x_2^2+\sqrt{64x_1^6+81x_1^2x_2^4}}{x_2^{5/3}x_1^{5/2}} \end{bmatrix}$$

```
> IsDefinite(Hf,'query'='negative_semidefinite');
  # Example illustrating use of IsDefinite command. Result demonstrates
  that H is negative semidefinite on domain, x1>0,x2>0.
```

$$0 \le \frac{490000}{9x_1x_2^{4/3}} \qquad 0 \le \frac{350}{9}\frac{8x_1^2+9x_2^2}{x_1^{3/2}x_2^{5/3}}$$

6.4.5 The Zero-Sum Matrix Game, Revisited

We end this section by revisiting the topic of matrix games from Section 4.1. Doing so will illustrate a connection between saddle points and duality theory.

Recall from Section 4.1 Steve and Ed's zero-sum matrix game involving the 3-by-3 *payoff matrix*

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 2 & 4 & -1 \\ -2 & 0 & 2 \end{bmatrix}. \tag{6.51}$$

In this game, at each play, Ed picks a column and, simultaneously, Steve picks a row. The dollar amount $a_{i,j}$ in the resulting entry then goes to Ed if it is positive and to Steve if it is negative. From Waypoint 4.1.3 we know that the mixed strategy Nash equilibrium for the game yields Ed and Steve's equilibrium mixed strategies, $\mathbf{x}_0$ and $\mathbf{y}_0$, respectively, along with their corresponding earnings. For Ed, $\mathbf{x}_0 = \begin{bmatrix} 3/28 \\ 9/28 \\ 4/7 \end{bmatrix}$, which indicates he should choose

column one with probability $\frac{3}{28}$, column two with probability $\frac{9}{28}$, and column three with probability $\frac{4}{7}$. Similarly, Steve's equilibrium mixed strategy of $\mathbf{y}_0 = \begin{bmatrix} 1/2 & 5/14 & 1/7 \end{bmatrix}$ dictates he should choose rows one through three with respective probabilities, $\frac{1}{2}$, $\frac{5}{14}$, and $\frac{1}{7}$. The game value corresponding to these strategies equals $\frac{13}{14}$. It represents the average amount Ed wins (and Steve loses) when each player follows his equilibrium mixed strategy. Because this game value and the equilibrium strategies constitute a mixed strategy Nash equilibrium, we know that Ed cannot increase his average earnings from $\frac{13}{14}$ by deviating from his equilibrium mixed strategy while Steve continues to follow his own. Likewise, we know that Steve cannot decrease his losses from $\frac{13}{14}$ by deviating from his equilibrium mixed strategy while Ed continues to follow his own.

In Section 4.1, we determined Ed's equilibrium mixed strategy, $\mathbf{x}_0$, was the solution of the following LP:

$$\begin{aligned} &\text{maximize } z \\ &\text{subject to} \\ &\qquad A\mathbf{x} \geq z\mathbf{e} \\ &\qquad \mathbf{e}^t \cdot \mathbf{x} = 1 \\ &\qquad \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{6.52}$$

(Recall that $\mathbf{e}$ denotes a 3-by-1 vector of 1s.) Steve's equilibrium mixed strategy, $\mathbf{y}_0$, was the solution of the dual LP corresponding to LP (6.52):

$$\begin{aligned} &\text{minimize } w \\ &\text{subject to} \\ &\qquad \mathbf{y}^t A \leq w\mathbf{e}^t \\ &\qquad \mathbf{e}^t \cdot \mathbf{y} = 1 \\ &\qquad \mathbf{y} \geq \mathbf{0}. \end{aligned} \tag{6.53}$$

(Note: Notation in this formulation of the dual LP differs slightly from that used in Section 4.1.5. Here we assume that $\mathbf{y}$ is a column vector, as opposed to a row vector, in $\mathbb{R}^3$.)

This zero-sum matrix game can also be described using results from this section. Consider the matrix product $\mathbf{y}^t A\mathbf{x}$. Since $x_1 + x_2 + x_3 = 1$ and $y_1 + y_2 + y_3 = 1$, this product defines a function $f : \mathbb{R}^4 \to \mathbb{R}$ as follows:

$$f(x_1, x_2, y_1, y_2) = \begin{bmatrix} y_1 \\ y_2 \\ 1 - y_1 - y_2 \end{bmatrix}^t A \begin{bmatrix} x_1 \\ x_2 \\ 1 - x_1 - x_2 \end{bmatrix}. \tag{6.54}$$

Waypoint 6.4.5. Show that f in (6.54) has one critical point, a saddle point, at

$$\begin{bmatrix} x_{1,0} \\ x_{2,0} \\ y_{1,0} \\ y_{2,0} \end{bmatrix} = \begin{bmatrix} 3/28 \\ 9/28 \\ 1/2 \\ 5/14 \end{bmatrix},$$

where $f(x_{1,0}, x_{2,0}, y_{1,0}, y_{2,0}) = \dfrac{13}{14}$. This demonstrates how the mixed strategy Nash equilibrium of a zero-sum matrix game corresponds to a saddle point of a single function, which is formed using the payoff matrix and whose output at the saddle point is the game value. That this outcome holds in general is a special case of classic result from game theory, known as the *von Neumann Minimax Theorem*.

We conclude this section by noting that, while neither player can improve his standing by unilaterally deviating from his equilibrium mixed strategy, it is possible for both players to simultaneously deviate from their equilibrium mixed strategies and for one or the other to benefit as a result. Exercise (8) illustrates this phenomenon.

Exercises Section 6.4

1. Determine the quadratic approximation,

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^t H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

   for the function $f(x_1, x_2) = e^{-\left(x_1^2 + x_2^2\right)}$ at $\mathbf{x}_0 = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$. Plot this approximation together with the function f.

2. Recall *Pam's Pentathlon Training Program*, Exercise 2, from Section 6.1. When stated as a maximization problem, the objective is represented by the function $f : S \to \mathbb{R}$, where $S = \mathbb{R}^3_+ = \{(x_1, x_2, x_3) \mid x_1, x_2, x_3 \geq 0\}$ and where

$$\begin{aligned} f(x_1, x_2, x_3) = .11193(254 - (180 - .5x_1 - x_2 - x_3))^{1.88} \\ + 56.0211(5.2 + .1x_1)^{1.05}. \end{aligned}$$

   In this formula, x_1, x_2, and x_3 denote the number of hours each week

Pam devotes to weight lifting, distance running, and speed workouts. The value of $f(x_1, x_2, x_3)$ represents the portion of Pam's total pentathlon score due to her performances in the 800 meter run and the shot put.

Show that the function f is strictly convex on S by establishing that its Hessian is positive definite there. (Hint: Express the Hessian in the form $H_f(\mathbf{x}) = \phi(\mathbf{x})A$, where ϕ is a positive function defined on S and where A is a positive definite matrix of numbers. Then each eigenvalue of $H_f(\mathbf{x})$ is a product of $\phi(\mathbf{x})$ and an eigenvalue of A.)

3. Calculate all critical points of each of the following functions on its stated domain S. Determine whether the function is convex, concave, or neither on S and then classify each critical point as a local or global minimum, local or global maximum, or saddle point.

   (a) $f(x_1, x_2) = x_1^2 + x_1x_2 - x_1 + 2x_2^2 - 4x_2 + 2, \quad S = \mathbb{R}^2$

   (b) $f(x_1, x_2) = x_1^2 - 10x_1x_2 + 4x_1 + 7x_2^2 - 8x_2 + 2, \quad S = \mathbb{R}^2$

   (c) $f(x_1, x_2) = -2x_1^2 + 6x_1x_2 - 6x_1 - 5x_2^2 + 8x_2 - 5, \quad S = \mathbb{R}^2$

   (d) $f(x_1, x_2) = x_1^4 - 8x_1^3 + 24x_1^2 - 32x_1 + 4x_2^2 - 4x_2, \quad S = \mathbb{R}^2$

   (e) $f(x_1, x_2) = \sin(x_1)\cos(x_2), \quad S = \{(x_1, x_2) \mid 0 < x_1, x_2 < \pi\}$

   (f) $f(x_1, x_2) = \ln(1 + x_1^2) + \dfrac{1}{2}x_2^2, \quad S = \{(x_1, x_2) \mid -1 < x_1 < 1\}$

   (g) $f(x_1, x_2) = e^{-\left(\frac{x_1^2+x_2^2}{2}\right)}, \quad S = \left\{(x_1, x_2) \mid x_1^2 + x_2^2 < 1\right\}$

4. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is the general quadratic mapping defined by

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} - \mathbf{b}^t\mathbf{x} + c, \tag{6.55}$$

   where A is a nonsingular, n-by-n matrix, $\mathbf{b}$ is in $\mathbb{R}^n$, and c is a real number. Show that $\mathbf{x} = A^{-1}\mathbf{b}$ is the sole critical point of f and that this critical point is a global minimum (resp. global maximum) if A is positive definite (resp. negative definite). Use this result to determine the global minimum of f in (6.55) when $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} -1 \\ 4 \end{bmatrix}$, and $c = 6$.

5. The general equation of a plane in $\mathbb{R}^3$ is given by $\mathbf{n}^t\mathbf{x} = d$, where $\mathbf{n}$ is a fixed vector in $\mathbb{R}^3$ (the *normal vector*), $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, and d is a constant. Determine a formula in terms of $\mathbf{n}$ for the point, $\mathbf{x}_0$, on this plane that is closest to the origin. Then find the point on the plane $x_1 + 2x_2 + 3x_3 = 1$ that is closest to the origin.

6. Show that the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by $f(x_1, x_2) = x_1^4 + x_2^2$ has a critical point at the origin, which is a global minimum. Then verify $H_f(\mathbf{0})$ is positive semidefinite. Then show that $f(x_1, x_2) = -x_1^4 + x_2^2$ has a critical point at the origin, which is a saddle point, and verify that $H_f(\mathbf{0})$ is still positive semidefinite. This example illustrates how, in the hypothesis of part (2) of Theorem 6.4.2, we may not replace "positive definite" with "positive semidefinite."

7. Give an example of a twice-differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$ having a global minimum at $\mathbf{x}_0 = \mathbf{0}$ and for which zero is the only eigenvalue of $H_f(\mathbf{x}_0)$. Then give an example of a twice-differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$ having a global maximum at $\mathbf{x}_0 = \mathbf{0}$ and for which zero is the only eigenvalue of $H_f(\mathbf{x}_0)$.

8. Consider the zero-sum matrix given involving the 2-by-2 payoff matrix

$$A = \begin{bmatrix} 2 & -3 \\ -1 & 4 \end{bmatrix}. \tag{6.56}$$

Suppose player 1 chooses columns one and two with respective probabilities, x_1 and x_2. Player 2 chooses rows one and two with respective probabilities y_1 and y_2.

   (a) Formulate the LP and dual LP, whose respective solutions indicate each player's optimal mixed strategy. Solve each of these LPs and determine the corresponding game value.

   (b) Confirm that the solution just obtained coincides with the saddle point of the function

$$f(x_1, y_1) = \begin{bmatrix} y_1 \\ 1 - y_1 \end{bmatrix}^t A \begin{bmatrix} x_1 \\ 1 - x_1 \end{bmatrix}.$$

   (c) Sketch the set of points in the $x_1 y_1$ plane that determine all possible mixed strategies for the two players that yield the game value from (a).

   (d) Now sketch the set of points in the $x_1 y_1$ plane that determine all possible mixed strategies for the two players that yield a game value that is 20% higher than that in (c).

9. Consider the zero-sum matrix game having the 2-by-2 payoff matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \tag{6.57}$$

Assume $a + d - (b + c) \neq 0$. Show that for the game to be fair, i.e., to have a game value of zero, the matrix A must be noninvertible.

Chapter 7

Numeric Tools for Unconstrained NLPs

7.1 The Steepest Descent Method

For many unconstrained NLPs, critical points of the objective function are difficult to compute using algebraic means. In this chapter, we investigate iterative numeric methods for overcoming this obstacle.

During the past several years, numeric methods have evolved and improved dramatically. We will not attempt to describe all developments in this field but will instead focus on three elementary tools: the *Steepest Descent Method*, *Newton's Method*, and the *Levenberg-Marquardt Algorithm*.

7.1.1 Method Derivation

Throughout this chapter, we assume $S \subseteq \mathbb{R}^n$ is open, $f : S \to \mathbb{R}$, and that our goal is to solve the unconstrained NLP,

$$\text{minimize } f(\mathbf{x}), \qquad \mathbf{x} \in S. \tag{7.1}$$

For a differentiable function, the most elementary method of estimating a solution to (7.1) is the *Steepest Descent Method*. As the term "descent" suggests, it approximates the minimum of a function f. However, when applied to $-f$, it approximates a maximum as well.

Assume $\mathbf{x}_0$ belongs to S, and recall from Exercise 3 of Section 6.2 that if $\mathbf{d}$ is a unit vector in $\mathbb{R}^n$, then the directional derivative of f at $\mathbf{x}_0$ in the direction of $\mathbf{d}$ is given by

$$\begin{aligned} f'_{\mathbf{d}}(\mathbf{x}_0) &= \lim_{h \to 0^+} \frac{f(\mathbf{x}_0 + h\mathbf{d}) - f(\mathbf{x}_0)}{h} \\ &= \nabla f(\mathbf{x}_0)^t \mathbf{d}. \end{aligned}$$

The Cauchy-Schwartz Inequality dictates that

$$\begin{aligned} \left|\nabla f(\mathbf{x}_0)^t \mathbf{d}\right| &\leq \|\nabla f(\mathbf{x}_0)\|\|\mathbf{d}\| \\ &= \|\nabla f(\mathbf{x}_0)\|, \end{aligned}$$

with equality in the absolute value possible if and only if $\mathbf{d} = \pm\dfrac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|}$. (See **??**.) Consequently $f'_{\mathbf{d}}(\mathbf{x}_0)$ is a minimum when $\mathbf{d}$ is chosen to have direction exactly opposite that of $\nabla f(\mathbf{x}_0)$. We summarize this fact by saying $-\nabla f(\mathbf{x}_0)$ is the *optimal descent direction* of f at x_0.

The Steepest Descent Method begins with an initial value $\mathbf{x}_0$ and a corresponding optimal descent direction, $-\nabla f(\mathbf{x}_0)$. If we define

$$\phi(t) = f\left(\mathbf{x}_0 - t\nabla f(\mathbf{x}_0)\right), \tag{7.2}$$

then ϕ is a decreasing function for t sufficiently small and positive.

For example, suppose that $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$, and $c = 6$, and that $f : \mathbb{R}^2 \to \mathbb{R}$ is the positive definite quadratic function given by

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^t A\mathbf{x} + \mathbf{b}^t\mathbf{x} + c \\ &= x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6. \end{aligned} \tag{7.3}$$

If $\mathbf{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, straightforward calculations show that $f(\mathbf{x}_0) = 12.5$ and $\nabla f(\mathbf{x}_0) = \begin{bmatrix} 2 \\ -6 \end{bmatrix}$. In this case, (7.2) yields $\phi(t) = 46t^2 - 40t + 12.5$.

Figures 7.1 and 7.2 provide two depictions of the function ϕ. Figure 7.1 illustrates the graph of f, along with an embedded arc consisting of the ordered triples

$$\left\{ \begin{bmatrix} \mathbf{x}_0 - t\nabla f(\mathbf{x}_0) \\ \phi(t) \end{bmatrix} \;\middle|\; 0 \le t \le 1 \right\}. \tag{7.4}$$

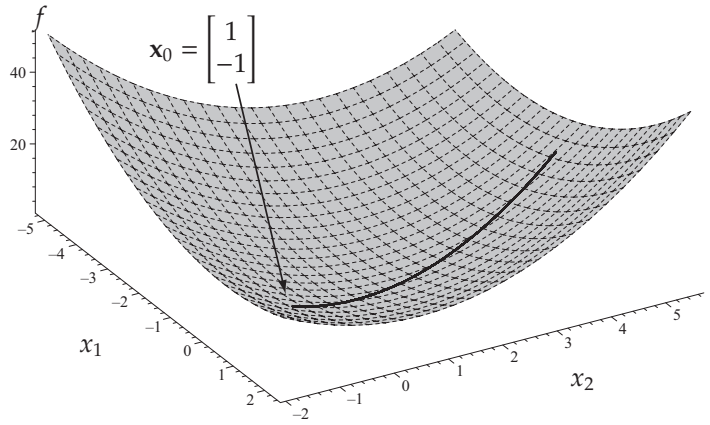


FIGURE 7.1: One depiction of $\phi(t) = f(\mathbf{x}_0 - t\nabla f(\mathbf{x}_0))$.

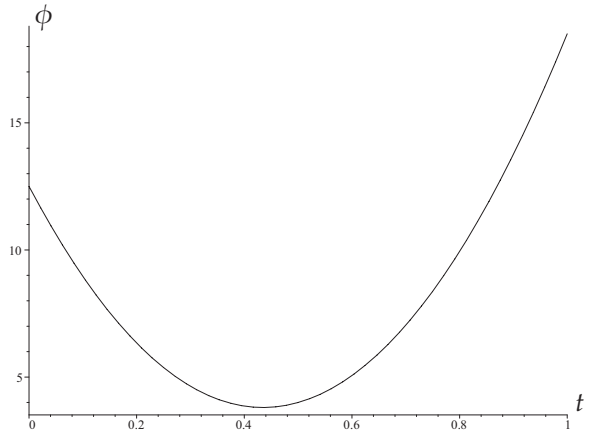


FIGURE 7.2: A plot of $\phi(t) = f(\mathbf{x}_0 - t\nabla f(\mathbf{x}_0))$.

Figure 7.2 depicts ϕ itself plotted as a function of t. The strict global minimum of ϕ is given by $t_0 = \dfrac{10}{23} \approx .435$. Using t_0, we define

$$\mathbf{x}_1 = \mathbf{x}_0 - t_0\nabla f(\mathbf{x}_0) = \begin{bmatrix} 3/23 \\ 37/23 \end{bmatrix} \approx \begin{bmatrix} .1304 \\ 1.609 \end{bmatrix}. \tag{7.5}$$

Note that $f(\mathbf{x}_1) = \phi(t_0) < \phi(0) = f(\mathbf{x}_0)$. In fact, direct computation shows that $f(\mathbf{x}_1) = 3.804$.

Thus, through a process of minimizing a single-variable function, we have obtained from $\mathbf{x}_0$ a new value $\mathbf{x}_1$, whose displacement from $\mathbf{x}_0$ is in the direction of steepest descent of f at $\mathbf{x}_0$, and whose corresponding objective function value is significantly less. That this process can be repeated using $\mathbf{x}_1$ as the new initial value leads to the creation of the algorithm known as the *Steepest Descent Method*, sometimes also referred to as Cauchy's Method, in honor of its inventor. We now summarize the steps of the algorithm.

The Steepest Descent Method
To obtain an approximate solution of NLP (7.1) under the assumption f is differentiable on S:

1. From the initial value $\mathbf{x}_0$ in S, compute the optimal descent direction, $\mathbf{d} = -\nabla f(\mathbf{x}_0)$.

2. Since S is open, for sufficiently small t, $\mathbf{x}_0 + t\mathbf{d}$ belongs to S. Calculate the smallest positive local minimum, t_0, of the function $\phi(t) = f(\mathbf{x}_0 + t\mathbf{d})$.

3. Using t_0 from (2), define $\mathbf{x}_1 = \mathbf{x}_0 + t_0\mathbf{d}$. Then proceed to (1) and replace $\mathbf{x}_0$ with $\mathbf{x}_1$. Compute the optimal descent direction at $\mathbf{x}_1$, a new single-variable function ϕ, and so on.

Repeated application of steps (1)-(3) generates a sequence of values

$$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$$

which, under ideal circumstances, converges to the global minimum of f on S. The decision to terminate the algorithm generally follows one of two approaches, each of which depends upon a specified tolerance, ϵ. The first involves applying the algorithm until successive approximations, $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, satisfy $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$. The second approach is to terminate the algorithm when $\left\|\nabla f(\mathbf{x}_k)\right\| < \epsilon$. Use of this latter inequality stems from the fact that at the global minimum, $\mathbf{x}_\bigstar$, of a differentiable function f, $\nabla f(\mathbf{x}_\bigstar) = \mathbf{0}$. Unless stated otherwise, we will always use the second of these two approaches for deciding when to terminate the algorithm.

Table 7.1 lists results of the Steepest Descent Method applied to the function $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$ using an initial value $\mathbf{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and a tolerance of $\epsilon = .01$. We elect to terminate the algorithm when $\left\|\nabla f(\mathbf{x}_k)\right\| < \epsilon$, which occurs after the ninth iteration. The ending value, to four significant digits, is given by $\mathbf{x}_9 = \begin{bmatrix} -1.397 \\ 1.796 \end{bmatrix}$. This is very close to the exact global minimum of f, $\mathbf{x}_\bigstar = \begin{bmatrix} -\frac{7}{5} \\ \frac{9}{5} \end{bmatrix}$.

TABLE 7.1: Results of Steepest Descent Method applied to $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$

k	$\mathbf{x}_k^t$	$\|\nabla f(\mathbf{x}_k)\|$
0	$[1, -1]$	$2\sqrt{10}$
1	$[0.1304, 1.609]$	3.025
2	$[-0.9326, 1.255]$	1.231
3	$[-1.102, 1.763]$	0.5895
4	$[-1.309, 1.694]$	0.2394
5	$[-1.342, 1.793]$	0.1151
6	$[-1.382, 1.779]$	0.04743
7	$[-1.389, 1.799]$	0.02247
8	$[0.004646, 0.00001]$	0.02077
9	$[-1.397, 1.796]$	0.009220

7.1.2 A Maple Implementation of the Steepest Descent Method

Maple's basic programming structures are well suited for implementing the Steepest Descent Method. The worksheet **Steepest Descent Method.mw** demonstrates one means for doing so, creating a procedure, `SteepestDescentMethod`, which approximates the minimum of a function of two variables, f. The procedure call takes the form

`SteepestDescentMethod(`*`function`*`,`*`initial`*`,`*`N`*`,`*`tolerance`*`)`,

where *`function`* is a differentiable function to be minimized, *`initial`* is the initial value, $\mathbf{x}_0$, (written as a list), *`N`* is the maximum number of iterations, and *`tolerance`* is the specified tolerance. The algorithm terminates when either the number of iterations is reached or the gradient norm is less than the desired tolerance. The returned value is the last iterate. Intermediate iterates are stored in a list of points, `iterates`, a global variable accessible outside of the procedure.

The following worksheet constructs the procedure and approximates the minimum of $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$ using an initial value of $\mathbf{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $N = 10$, and a tolerance of .01. (Note: Because this worksheet uses the `LinearAlgebra` package version of the `Norm` command, it is important to load the `LinearAlgebra` package after loading `VectorCalculus`.)

```
> restart: with(VectorCalculus):with(LinearAlgebra):
> SteepestDescentMethod := proc (function, initial, N, tolerance)
  local x,f,Delf,d,j,epsilon,phi,t0; global iterates:
  # Create local and global variables.
  x:=array(0 .. N):
```

```
  # Define array of iterates.
  f:=unapply(function, [x1, x2]):
  # Create function.
  Delf:=unapply(Gradient(function, [x1, x2]), [x1, x2]):
  # Create corresponding gradient function.
  x[0]:=evalf(initial):
  # Set initial array value.
  epsilon := 10:
  # Set initial epsilon to a large value.
  j:=0: while (tolerance <= epsilon and j<=N-1) do
  # Create loop structure to perform algorithm.
  d:=-Delf(op(x[j])): # Compute optimal descent direction.
  phi:=unapply(simplify(f(x[j][1]+t*d[1],x[j][2]+t*d[2])),t):
  t0:=subs(fsolve(D(phi)(t) = 0, {t = 0..infinity}, maxsols = 1),t):
  # Determine minimum of phi.
  x[j+1]:=[x[j][1]+t0*d[1],x[j][2]+t0*d[2]]:
  # Use minimum of phi to construct next iterate.
  epsilon:=evalf(Norm(Delf(x[j+1][1],x[j+1][2]),Euclidean)):
  # Update epsilon using gradient.
  j:=j+1:
  # Increase loop index.
  end do:
  iterates:=[seq(x[i],i=0..j)]:RETURN(x[j]):end:
  # Construct iterates and return last array value.
> f:=(x1,x2)->x1^2+x1*x2+3/2*x2^2+x1-4*x2+6;
  # Enter function.
```

$$f := (x_1, x_2) \to x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$$

```
> SteepestDescentMethod(f(x1, x2), [1, -1], 10, .01);
  # Procedure call using initial value [1,-1], bound of 10 iterations,
  and tolerance, .01.
```

$$[-1.396540715, 1.795964168]$$

```
> iterates;
  # Print iterates.
```

$$[[1, -1], [.1304347826, 1.608695652], [-.9323671494), 1.254428341],$$
$$[-1.101799342), 1.762724918], [-1.308883132), 1.693696988],$$
$$[-1.341896490), 1.792737062], [-1.382246149), 1.779287175],$$
$$[-1.388678705), 1.798584838], [-1.396540715), 1.795964168]]$$

Note that the list, `iterates`, can then be plotted using the `pointplot` command.

7.1.3 A Sufficient Condition for Convergence

Of course, the preceding worksheet illustrates an ideal situation, in which the sequence of iterates obtained using the Steepest Descent Method converges to a global minimum. The following Waypoint illustrates how this is not always the case in general.

◆ ——— ◆

Waypoint 7.1.1. Apply the Steepest Descent Method to the function, $f : \mathbb{R}^2 \to \mathbb{R}$, defined by $f(x_1, x_2) = x_1^4 + x_2^4 - 4x_1x_2 + 1$. Use a tolerance of $\epsilon = .1$, and investigate the outcome using three different initial values, $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\mathbf{x}_0 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$, and $\mathbf{x}_0 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$. Compare your results with those obtained using algebraic methods from Section 6.4.

◆ ——— ◆

The preceding example, and others like it, demonstrate how the Steepest Descent Method can generate a sequence, $\{\mathbf{x}_k\}$, that converges to a critical point of f that is a local, but not global, minimum or that is a saddle point. Thus, additional conditions must be met in order to ensure that the sequence of iterates converges in the first place, and, if it does, whether it yields a local or global minimum.

General conditions on f under which a sequence of iterates leads to critical points are provided by a special case of a theorem due to Zoutendijk. We shall omit the proof of this result, which can be found in more advanced texts [36],[48].

Theorem 7.1.1. Assume $S \subseteq \mathbb{R}^n$ is nonempty and open, and suppose $f : S \to \mathbb{R}$ is differentiable on S. Choose $\mathbf{x}_0$ belonging to S, and define the *lower level set*,

$$S_0 = \{\mathbf{x} \mid \mathbf{x} \in S \text{ and } f(\mathbf{x}) \leq f(\mathbf{x}_0)\}.$$

Suppose that ∇f is *Lipschitz continuous* in an open set O containing S_0, meaning there exists a constant K (referred to as the *Lipschitz constant*) such that

$$\left\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\right\| \leq K\|\mathbf{x}_1 - \mathbf{x}_2\| \text{ for all } \mathbf{x}_1, \mathbf{x}_2 \in O.$$

If there exists $m \in \mathbb{R}$ such that $m \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$, then the sequence $\{\mathbf{x}_k\} \subseteq S_0$ generated using the Steepest Descent Method satisfies

$$\lim_{k\to\infty} \left\|\nabla f(\mathbf{x}_k)\right\| = 0. \tag{7.6}$$

This theorem does not state that the sequence $\{\mathbf{x}_k\}$ converges to the global minimum of f on S. In fact, it does not even state that this sequence converges

at all. However, if even some subsequence of this sequence has a limit point, $\mathbf{x}_\bigstar$, in S, then the Lipschitz continuity and (7.6) guarantee $\nabla f(\mathbf{x}_\bigstar) = \mathbf{0}$. (We note that if S_0 is closed and bounded, then such a subsequence must exist due to a fundamental analysis result known as the *Bolzano-Weierstrass Theorem*. The limit of the subsequence must belong to S_0 or its boundary.) Because $f(\mathbf{x}_k)$ is decreasing in k, $\mathbf{x}_\bigstar$ must therefore correspond to either a local minimum or a saddle point.

Verifying that ∇f is Lipschitz continuous by definition can be tedious and should be avoided when possible. For several classes of functions, however, Lipschitz continuity follows from certain *matrix norm* inequalities. (See Appendix B.) Three straightforward cases are the following:

1. If $f : \mathbb{R}^n \to \mathbb{R}$ is the general quadratic function defined by $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} + \mathbf{b}^t\mathbf{x} + c$, where A is an n by n matrix, $\mathbf{b}$ belongs to $\mathbb{R}^n$, c is real, then $\nabla f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ so that
$$\begin{aligned}\left\|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\right\| &= \|A\left(\mathbf{x}_1 - \mathbf{x}_2\right)\| \\ &\leq \|A\|\|\mathbf{x}_1 - \mathbf{x}_2\|,\end{aligned}$$
Here, $\|A\|$ denotes the *spectral norm* of A, defined as the square root of the largest eigenvalue of AA^t.

2. If f is as defined as above and A is also symmetric, eigenvalues of AA^t are the eigenvalues of A^2, and the spectral norm reduces to $\|A\| = \rho(A)$, where $\rho(A)$ is the *spectral radius*, or maximum of the absolute values of the eigenvalues of A.

3. More generally, if f all second-order partial derivatives of f exist and are continuous on O, then ∇f is Lipschitz continuous on O provided $\|H_f(\mathbf{x})\| = \rho\left(H_f(\mathbf{x})\right)$ is bounded by a constant independent of $\mathbf{x}$ in O. We will omit the proof of this result, which follows from a generalization to $\mathbb{R}^n$ of the Mean Value Theorem[4].

The unconstrained *ConPro Manufacturing Company* NLP,

$$\begin{aligned}\text{minimize} f(x_1, x_2) &= -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2, \\ \text{where } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &\in S = \mathbb{R}^2_+ = \{(x_1, x_2) \mid x_1, x_2 > 0\}, \quad (7.7)\end{aligned}$$

illustrates the usefulness of Theorem 7.1.1. (Hereafter, we will state the *ConPro* objective in terms of minimization.) We first note that f is bounded below on S. Now suppose we let $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, which corresponds to the lower level set $S_0 = \{\mathbf{x} \mid f(\mathbf{x}) \leq f(10, 10)\}$ shown in Figure 7.3.

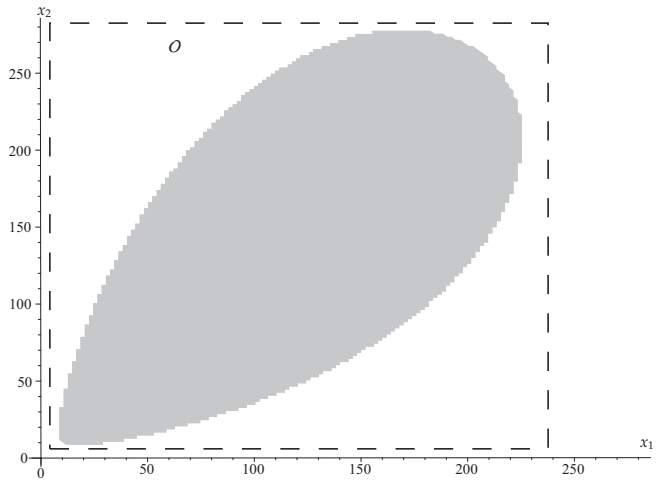


FIGURE 7.3: The lower level set $S_0 = \{\mathbf{x} \mid f(\mathbf{x}) \leq f(10, 10)\}$.

Clearly we can draw an bounded, open set O in S that contains S_0 and has the property that for any $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ in O, $\delta < x_1, x_2 \leq M$ for some $\delta > 0$ and finite M. The set contained within the dashed lines in Figure 7.3 is one such example.

The eigenvalues of the Hessian of f are given by

$$\lambda = \frac{175\left(9x_1x_2^2 + 8x_1^3 \pm \sqrt{81x_2^4x_1^2 + 64x_1^6}\right)}{9x_1^{\frac{5}{2}}x_2^{\frac{5}{3}}}, \tag{7.8}$$

which are in fact both positive. The larger of these two eigenvalues is obtained by using the positive root in (7.8). This fraction is bounded by a constant independent of $\mathbf{x}$ in O, due to the fact that its numerator is bounded there and its denominator is bounded away from zero. Specifically,

$$\begin{aligned} |\lambda| &= \frac{175\left(9x_1x_2^2 + 8x_1^3 + \sqrt{81x_2^4x_1^2 + 64x_1^6}\right)}{9x_1^{\frac{5}{2}}x_2^{\frac{5}{3}}} \\ &\leq \frac{175\left(9MM^2 + 8M^3 + \sqrt{81M^4M^2 + 64M^6}\right)}{9\delta^{\frac{5}{2}}\delta^{\frac{5}{3}}} \\ &= \frac{175\left(17M^3 + \sqrt{145M^6}\right)}{9\delta^{\frac{25}{6}}}. \end{aligned}$$

Thus, ∇f is Lipschitz continuous on O, and we conclude that the hypothesis of Theorem 7.1.1 is satisfied. The first eight Steepest Descent Method iterates,

starting from $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, are given in Table 7.2. In the fourth column, $\|\mathbf{x}_k - \mathbf{x}_\star\|$ measures the distance from $\mathbf{x}_k$ to the true minimum of f, which we computed in Section 6.4 and is given by

$$\mathbf{x}_\star = \left(\frac{784}{9}, \frac{2744}{27}\right) \approx (87.11, 101.63).$$

TABLE 7.2: Results of the Steepest Descent Method applied to the *ConPro* objective function, f, from (7.7)

k	$\mathbf{x}_k^t$	$\|\nabla f(\mathbf{x}_k)\|$	$\|\mathbf{x}_k - \mathbf{x}_\star\|$
0	[10, 10]	173.245	119.759
1	[91.680, 85.907]	40.277	16.373
2	[82.493, 95.792]	3.6065	7.444
3	[87.236, 100.200]	2.788	1.435
4	[86.558, 100.930]	.420	.892
5	[87.125, 101.455]	.334	.175
6	[87.043, 101.543]	.0152	.110
7	[87.113, 101.608]	.0411	.022
8	[87.103, 101.619]	.006	.014

7.1.4 The Rate of Convergence

Theorem 7.1.1 provides conditions under which the Steepest Descent Method yields a sequence of iterates that converges to a critical point. It does not, however, convey information regarding rates of convergence. Theorem 7.1.2 serves this purpose.

Theorem 7.1.2. Suppose that $S \subseteq \mathbb{R}^n$ is open, that $f : S \to \mathbb{R}$ has continuous second-order partial derivatives, and that the Steepest Descent Method starting from $\mathbf{x}_0$ generates a sequence of iterates $\{\mathbf{x}_k\}$ converging to the local minimum of f, which we label $\mathbf{x}_\star$. If $H_f(\mathbf{x}_\star)$ is positive definite, then the objective values satisfy

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_\star) \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 (f(\mathbf{x}_k) - f(\mathbf{x}_\star)), \quad k = 0, 1, \ldots, \tag{7.9}$$

where λ_n and λ_1 denote the largest and smallest eigenvalues of $H_f(\mathbf{x}_\star)$, respectively.

Furthermore, if f is a positive definite quadratic function, meaning it takes the form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} + \mathbf{b}^t\mathbf{x} + c, \tag{7.10}$$

where c is real, $\mathbf{b}$ belongs to $\mathbb{R}^n$, and A is an n-by-n positive definite matrix, then the Steepest Descent Method starting from any $\mathbf{x}_0$ in $\mathbb{R}^n$ generates a sequence of inputs, $\{\mathbf{x}_k\}$, that converges to the global minimum of f, $\mathbf{x}_\bigstar$ at a rate governed by

$$\left\|\mathbf{x}_k - \mathbf{x}_\bigstar\right\|^2 \leq \frac{2}{\lambda_1}\left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^{2k}\left(f(\mathbf{x}_0) - f(\mathbf{x}_\bigstar)\right), \quad k = 0, 1, \ldots. \tag{7.11}$$

Inequality (7.9) is sometimes rephrased by stating that the sequence of objective values $\{f(\mathbf{x}_k)\}$ converges to $f(\mathbf{x}_\bigstar)$ at a *linear rate*. Use of this term stems from the fact that the ratio, $\dfrac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}_\bigstar)}{f(\mathbf{x}_k) - f(\mathbf{x}_\bigstar)}$, is bounded.

The proof of Theorem 7.1.2 can be found in more advanced texts [4]. Instead of proving it, we examine its validity in the context of the positive definite quadratic function from (7.3). Thus, $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} + \mathbf{b}^t\mathbf{x} + c$, where $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$, and $c = 6$.

Assume that $\mathbf{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, in which case $f(\mathbf{x}_0) = 12.5$. Recall that the global minimum is given by $\mathbf{x}_\bigstar = \begin{bmatrix} -\frac{7}{5} \\ \frac{9}{5} \end{bmatrix}$ and $f(\mathbf{x}_\bigstar) = 1.7$.

The Hessian, $H_f(\mathbf{x})$ is simply A, whose eigenvalues are given by $\lambda_1 = \dfrac{5 - \sqrt{5}}{2}$ and $\lambda_1 = \dfrac{5 + \sqrt{5}}{2}$. Thus

$$\left(\frac{\lambda_2 - \lambda_1}{\lambda_2 + \lambda_1}\right)^2 = \frac{1}{5}.$$

Using the first four Steepest Descent Method iterates from Table 7.1, we obtain the values shown in Table 7.3. In each row, the ratio of the second entry to the third is always less than or equal to $\frac{1}{5}$, thereby demonstrating the validity of inequality (7.9) for these first few iterates.

Table 7.4 focuses on the iterates themselves and not their corresponding objective values. It illustrates the validity of inequality (7.11).

Perhaps the most important conclusion we can draw from Theorem 7.1.2 is that eigenvalues play a crucial role in determining convergence rate. In particular, the greater the difference between the largest and smallest eigenvalues of $H_f(\mathbf{x}_\bigstar)$, the closer that $\left(\dfrac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)$ becomes to one. This in turn slows

TABLE 7.3: Objective output differences

k	$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_\star)$	$f(\mathbf{x}_k) - f(\mathbf{x}_\star)$
0	2.1043	10.8000
1	.4100	2.1043
2	.0799	.4100
3	.0156	.0799
4	.0030	.0156

TABLE 7.4: Error between $\mathbf{x}_k$ and $\mathbf{x}_\star$

k	$\left\|\mathbf{x}_k - \mathbf{x}_\star\right\|^2$	$\frac{2}{\lambda_1}\left(\frac{\lambda_n-\lambda_1}{\lambda_n+\lambda_1}\right)^{2k}(f(\mathbf{x}_0) - f(\mathbf{x}_\star))$
0	13.6000	15.6210
1	2.3788	3.1260
2	.5163	.6251
3	.0903	.1250
4	.0196	.0250

down the rates of convergence. Exercises 1 and 2 provide further examples illustrating this phenomenon.

Exercises Section 7.1

1. Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$ is defined by $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} + \mathbf{b}^t\mathbf{x}$, where $A = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$, and let $\mathbf{x}_0 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$.
   (a) Calculate the global minimum of f on $\mathbb{R}^2$.
   (b) Determine the optimal descent direction, $\mathbf{d}$, of f at $\mathbf{x}_0$.
   (c) Calculate the smallest positive local minimum of the function $\phi(t) = f(\mathbf{x}_0 + t\mathbf{d})$.
   (d) Use your previous result to calculate the first Steepest Descent Method iterate, $\mathbf{x}_1$.
   (e) Calculate the next four Steepest Descent Method iterates, and verify that formulas (7.9) and (7.11) are valid.
   (f) How many iterations are required before $\left\|\nabla f(\mathbf{x}_k)\right\| < .01$?
2. Repeat the previous question using $A = \begin{bmatrix} 10 & 0 \\ 0 & .1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$, and $\mathbf{x}_0 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$.

How does the rate of convergence to the global minimum compare with that in the previous question? What accounts for any major differences?

3. Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$ is defined by $f(x_1, x_2) = x_1^4 + 2x_2^4 + 5x_1^2 + 4x_2^2 - 10x_1^2x_2$. By choosing different initial values, demonstrate how the Steepest Descent Method leads to sequences converging to a local minimum, a global minimum, and a saddle point of f.

4. Assume that f is the *ConPro* objective function from (7.7) and that $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$. Verify that the first five Steepest Descent Method iterates satisfy inequality (7.9).

5. Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$ is defined by $f(x_1, x_2) = \ln(1 + x_1^2) + x_2^2$ and that $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$. Observe that f is bounded below by zero.

   (a) Plot the lower set,

   $$S_0 = \{\mathbf{x} \mid \mathbf{x} \in S \text{ and } f(\mathbf{x}) \leq f(\mathbf{x}_0)\}.$$

   Note: Maple's `implicitplot` command can assist in this process. If the function, f, has been defined and the `plots` package has been loaded, enter the following command:

   ```
   implicitplot(f(x1,x2)<=evalf(f(2,2)),x1=-20..20,x2=-5..5,
   coloring=[grey,white],filled=true);
   ```

   (b) Verify that there is an open set O containing S_0, in which $\rho\left(H_f(\mathbf{x})\right)$ is bounded.

   (c) Using $\mathbf{x}_0$ as an initial value, verify that the sequence of Steepest Descent Method iterates converges to the origin, which is in fact a global minimum.

6. Repeat the previous question using $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x_1, x_2) = x_1^4 - 8x_1^3 + 24x_1^2 - 32x_1 + 4x_2^2 - 4x_2$ and $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$.

7.2 Newton's Method

7.2.1 Shortcomings of the Steepest Descent Method

A major drawback of the Steepest Descent Method is its slow rate of convergence near the minimum. The simple positive definite quadratic function, $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$, from (7.3) illustrates this phenomenon, both in Table 7.1, where the relative change in $\left\|\nabla f\left(\mathbf{x}_k\right)\right\|$ is largest for the first few iterations, but also in Figure 7.4. Superimposed on this diagram are four contours of f, the first three Steepest Descent Method iterates, and scaled versions of the corresponding optimal descent directions, $-\nabla f(\mathbf{x}_k)$, where $k = 0, 1, 2, 3$. The descent direction vectors demonstrate a drawback with the Steepest Descent Method known as "zig-zagging" or "hemstitching." This phenomenon, which frequently results for functions having mildly "elliptic" or "elongated" valleys, slows the rate of convergence.

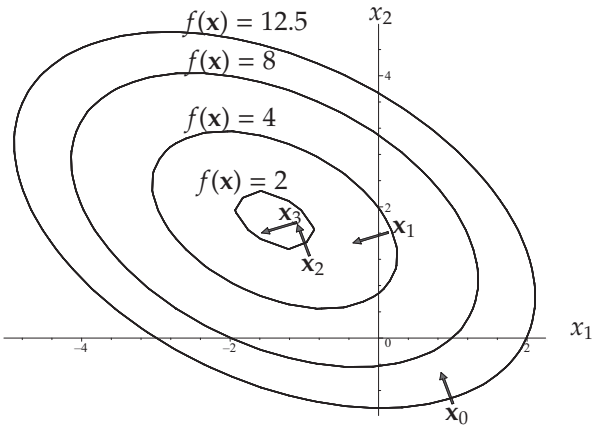


FIGURE 7.4: Steepest descent approximations x_k, $k = 0, 1, 2, 3$ for $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$, along with scaled optimal descent directions.

7.2.2 Method Derivation

Newton's Method is well-suited for addressing this problem, although it does require that the objective function, f, be twice-differentiable, and it has its own computational drawbacks. Suppose $S \subseteq \mathbb{R}^n$, $f : S \to \mathbb{R}$ is twice-differentiable, and $\mathbf{x}_0$ belongs to the interior of S. For all $\|\mathbf{d}\|$ sufficiently small, $\mathbf{x}_0 + \mathbf{d}$ belongs to S and second-order differentiability implies

$$f(\mathbf{x}_0) - f(\mathbf{x}_0 + \mathbf{d}) \approx -\nabla f(\mathbf{x}_0)^t \mathbf{d} - \frac{1}{2}\mathbf{d}^t H_f(\mathbf{x}_0)\mathbf{d}. \tag{7.12}$$

If we intend to express the first iterate, $\mathbf{x}_1$, in the form $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{d}$, then the decrease in f is maximized by differentiating the right-hand side of (7.12) with respect to the vector $\mathbf{d}$. Doing so, setting the result equal to $\mathbf{0}$ and solving for $\mathbf{d}$, we obtain

$$\mathbf{d} = -H_f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0), \tag{7.13}$$

provided $H_f(\mathbf{x}_0)^{-1}$ exists.

We call the vector $\mathbf{d}$ from (7.13) the *Newton direction* of f at $\mathbf{x}_0$. It is the second-order analog of the optimal descent direction associated with the Steepest Descent Method. Like the optimal descent direction, it produces a curve, "embedded" in the graph of f. Figure 7.5 depicts both the optimal descent and Newton directions starting from $\mathbf{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ for the quadratic function, $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$.

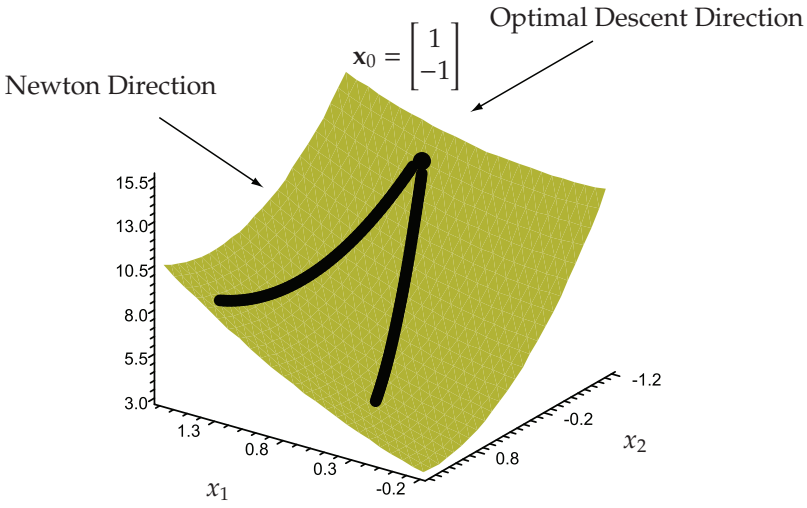


FIGURE 7.5: Illustration of optimal descent and Newton directions for $f(x_1, x_2) = x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6$ starting at $\mathbf{x}_0$.

Thus, starting from $\mathbf{x}_0$ and moving in the Newton direction leads to $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{d} = \mathbf{x}_0 - H_f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0)$. Of course, assuming invertibility of the Hessian at each stage, this process can be repeated, thereby leading to Newton's Method. We now summarize the steps of the algorithm.

Newton's Method
To obtain an approximate solution of NLP (7.1) under the assumption f is twice-differentiable on S:

1. From the initial value $\mathbf{x}_0$ in S, calculate the Newton direction, $\mathbf{d} = -H_f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0)$, provided $H_f(\mathbf{x}_0)^{-1}$ exists.
2. Define $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{d} = \mathbf{x}_0 - H_f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0)$.
3. Proceed to (1) and replace $\mathbf{x}_0$ with $\mathbf{x}_1$. Compute the Newton direction of f at $\mathbf{x}_1$, $\mathbf{x}_2$, and so on.

For example, recall the positive definite quadratic function from (7.3). If $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$, and $c = 6$, this function is given by

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^t A\mathbf{x} - \mathbf{b}^t\mathbf{x} + c \\ &= x_1^2 + x_1x_2 + \frac{3}{2}x_2^2 + x_1 - 4x_2 + 6 \end{aligned}$$

If $\mathbf{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, then $\nabla f(\mathbf{x}_0) = \begin{bmatrix} 2 \\ -6 \end{bmatrix}$ and $H_f(\mathbf{x}_0) = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$. Thus, the Newton direction at $\mathbf{x}_0$ becomes $\mathbf{d} = \begin{bmatrix} -2.4 \\ 2.8 \end{bmatrix}$ and $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{d} = \begin{bmatrix} -1.4 \\ 1.8 \end{bmatrix}$. Recall from Section 7.1 that $\mathbf{x}_1$ is precisely the global minimum of f! That the first iterate of Newton's Method leads to a global minimum is true in general for all positive definite quadratic mappings. Verification of this fact is fairly straightforward.

◆ ——— ◆

Waypoint 7.2.1. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is the positive definite quadratic function defined by

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} - \mathbf{b}^t\mathbf{x} + c,$$

where A is any n-by-n positive definite matrix, $\mathbf{b}$ is in $\mathbb{R}^n$, and c is a real scalar.

1. Compute $\nabla f(\mathbf{x})$, and use your result to determine the global minimum $\mathbf{x}_\bigstar$ in terms of A and $\mathbf{b}$.
2. Given $\mathbf{x}_0$ in $\mathbb{R}^n$, determine the Newton direction in terms of $\mathbf{x}_0$, A, and $\mathbf{b}$.
3. Show that the first iterate, $\mathbf{x}_1$, obtained using Newton's method is the global minimum obtained in (1).

◆ ——— ◆

7.2.3 A Maple Implementation of Newton's Method

The `SteepestDescentMethod` procedure outlined in Section 7.1 is easily modified to create a new procedure, `NewtonsMethod`, which may be found in the worksheet **Newtons Method.mw**. The procedure call takes the form

`NewtonsMethod(`*`function`*`,`*`initial`*`,`*`N`*`,`*`tolerance`*`)`,

where *`function`* is a twice-differentiable function to be minimized, *`initial`* is the initial value, $\mathbf{x}_0$, (written as a list), *`N`* is the maximum number of iterations, and *`tolerance`* is the specified tolerance. The algorithm terminates when either the number of iterations is reached or the gradient norm is less than the desired tolerance. The returned value is the approximate minimum value. Intermediate iterates leading to the approximate minimum are stored in a list, `iterates`, a global variable accessible outside of the procedure.

The following worksheet constructs the procedure and approximates the minimum of the *ConPro* objective, $f(x_1, x_2) = -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2$ using $\mathbf{x}_0 = \begin{bmatrix}10\\10\end{bmatrix}$, $N = 5$, and a tolerance of $\epsilon = .01$.

```
> restart: with(VectorCalculus):with(LinearAlgebra):
> NewtonsMethod := proc (function, initial, N, tolerance)
  local x,f,Delf,Hf,d,j,epsilon; global iterates:
  # Create local and global variables.
  x:=array(0 .. N):
  # Define array of iterates.
  f:=unapply(function, [x1, x2]):
  # Create function.
  Delf:=unapply(Gradient(function, [x1, x2]), [x1, x2]):
  # Create corresponding gradient function.
  Hf:=unapply(Hessian(function, [x1, x2]), [x1, x2]):
  # Create corresponding Hessian matrix function.
  x[0]:=evalf(initial):
  # Set initial array value.
  epsilon := 10:
  # Set initial epsilon to a large value.
  j:=0: while (tolerance <= epsilon and j<=N-1) do
  # Create loop structure to perform algorithm.
  d:=-convert(MatrixInverse(Hf(x[j][1],x[j][2])).Delf(x[j][1],x[j][2]),list):
  # Compute Newton direction.
  x[j+1]:=[x[j][1]+d[1],x[j][2]+d[2]]:
  # Compute next iterate.
  epsilon:=evalf(Norm(Delf(x[j+1][1],x[j+1][2]),Euclidean)):
  # Update epsilon using gradient.
```

```
    j:=j+1:
    # Increase loop index.
    end do:
    iterates:=[seq(x[i],i=0..j)]:RETURN(x[j]):end:
    # Construct iterates and return last array value.
> f:=(x1,x2)->-1400x1^(1/2)x2^(1/3)+350x1+200x2;
    # Enter function
```

$$f := (x_1, x_2) \to -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2$$

```
> NewtonsMethod(f(x1, x2), [10,10], 5, .01);
    # Procedure call using initial value [10,10], bound of 5 iterations,
    and tolerance, .01.
```

$$[87.10975551, 101.6262013]$$

```
> iterates;
    # Print iterates.
```

$$[[10., 10.], [28.06287805, 29.11130610], [56.81932208, 61.72021291],$$
$$[80.07576281, 91.03969995], [86.73692387, 100.9216310], [87.10975551, 101.6262013]]$$

7.2.4 Convergence Issues and Comparison with the Steepest Descent Method

The preceding examples suggest that Newton's Method is more effective for approximating a local or global minimum of a function than is the Steepest Descent Method. However, this result is not true in general, and Newton's Method has its own computation drawbacks. Among these are the following:

1. Newton's Method assumes second-order differentiability of the objective function, f, and requires evaluating the Hessian matrix, $H_f(\mathbf{x}_k)$, at each iteration.

2. Whereas the Steepest Descent Method requires solving an equation in a single variable at each iteration, Newton's Method requires solving a system of equations. This task frequently requires a much greater number of computations.

3. To obtain a unique solution to this system, we require that $H_f(\mathbf{x}_k)$ be invertible, which may not be the case. As a result, the method fails altogether. However, even if $H_f(\mathbf{x}_0)$ is invertible, it may be "almost non-invertible," as measured by having eigenvalues extremely small in magnitude. Numeric results then become very sensitive to computational roundoff error, a general phenomenon known as *ill-conditioning*.

While Newton's Method has its drawbacks, the advantage of its use stems from the fact that an initial value sufficiently close to a known local minimum yields a sequence of iterates converging to the minimum more rapidly to the minimum than a sequence obtained using the Steepest Descent Method. Theorem 7.2.1 summarizes this result.

The statement of this theorem requires us to recall the notion of a *matrix norm* as summarized in Appendix **??**. Any matrix norm, $\|\cdot\|$, on $\mathbb{R}^n$ satisfies $\|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|$ for every $\mathbf{x}$ in $\mathbb{R}^n$. As our discussion focuses almost exclusively on the case when A is symmetric, we will use, $\|A\| = \rho(A)$, where $\rho(A)$ is the spectral radius, or maximum of the absolute values of the eigenvalues of A.

Because the Hessian matrix, $H_f(\mathbf{x})$, is a function of $\mathbf{x}$, so too will be its matrix norm. We say that $H_f(\mathbf{x})$ is Lipschitz continuous in the matrix norm if there exists a constant, M, satisfying

$$\left\|H_f(\mathbf{x}_1) - H_f(\mathbf{x}_2)\right\| \leq M\|\mathbf{x}_1 - \mathbf{x}_2\|. \tag{7.14}$$

Phrased another way, the norm of the matrix difference, $H_f(\mathbf{x}_1) - H_f(\mathbf{x}_2)$, is bounded by a constant times the distance from $\mathbf{x}_2$ to $\mathbf{x}_1$. A consequence of (7.15), whose proof we omit, is that if $H_f(\mathbf{x}_1)$ is positive definite and if $H_f(\mathbf{x})$ is Lipschitz continuous in some neighborhood of $\mathbf{x}_1$, then $H_f(\mathbf{x}_2)$ is also positive definite and bounded below in norm by some constant, m, independent of $\mathbf{x}_2$, provided $\|\mathbf{x}_1 - \mathbf{x}_2\|$ is sufficiently small.

With this background in mind, we now state an important convergence result.

Theorem 7.2.1. Assume $S \subseteq \mathbb{R}^n$ is nonempty, open set. Suppose $f : S \to \mathbb{R}^n$ is continuously twice-differentiable, having a local minimum at $\mathbf{x}_\bigstar$, where $H_f(\mathbf{x}_\bigstar)$ is positive definite. Suppose that for $\mathbf{x}$ sufficiently close to $\mathbf{x}_\bigstar$, $H_f(\mathbf{x})$ is Lipschitz continuous in the matrix norm, meaning there exists a constant, M, satisfying

$$\left\|H_f(\mathbf{x}_1) - H_f(\mathbf{x}_2)\right\| \leq M\|\mathbf{x}_1 - \mathbf{x}_2\|, \tag{7.15}$$

for all $\mathbf{x}_1$ and $\mathbf{x}_2$ sufficiently close to $\mathbf{x}_\bigstar$. Then, if $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}_\bigstar$, the sequence of Newton Method iterates satisfies

$$\left\|\mathbf{x}_{k+1} - \mathbf{x}_\bigstar\right\| \leq C\left\|\mathbf{x}_k - \mathbf{x}_\bigstar\right\|^2 \text{ for } k = 0, 1, 2, \ldots, \tag{7.16}$$

for some constant, C.

Proof. Choose δ in $(0, 1)$ small enough to satisfy the following three conditions:

1. Inequality (7.15) holds whenever $\left\|\mathbf{x}_1 - \mathbf{x}_\bigstar\right\|$ and $\left\|\mathbf{x}_2 - \mathbf{x}_\bigstar\right\|$ are less than δ.

2. $H_f(\mathbf{x})$ is positive definite if $\|\mathbf{x} - \mathbf{x}_\bigstar\| < \delta$

3. $0 < m \leq \|H_f(\mathbf{x})\|$ if $\|\mathbf{x} - \mathbf{x}_\bigstar\| < \delta$

Suppose $\mathbf{x}_0$ is chosen close enough to $\mathbf{x}_\bigstar$ so that

$$\|\mathbf{x}_0 - \mathbf{x}_\bigstar\| < \min\left\{\delta, \sqrt{\frac{2m}{M}} \cdot \delta\right\}. \tag{7.17}$$

.

By (2), $H_f(\mathbf{x})$ is invertible so that $\mathbf{x}_1 = \mathbf{x}_0 - H_f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0)$ is defined. Since $\nabla f(\mathbf{x}_\bigstar) = \mathbf{0}$, we have

$$\begin{aligned}\mathbf{x}_1 - \mathbf{x}_\bigstar &= \mathbf{x}_0 - H_f(\mathbf{x}_0)^{-1}\nabla f(\mathbf{x}_0) - \mathbf{x}_\bigstar \\ &= \mathbf{x}_0 - \mathbf{x}_\bigstar + H_f(\mathbf{x}_0)^{-1}\left(\nabla f(\mathbf{x}_\bigstar) - \nabla f(\mathbf{x}_0)\right).\end{aligned} \tag{7.18}$$

We now focus on the difference $(\nabla f(\mathbf{x}_\bigstar) - \nabla f(\mathbf{x}_0))$ in (7.18) and introduce a vector-valued function $\phi : [0,1] \to \mathbb{R}^n$ given by

$$\phi(t) = \nabla f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right).$$

Note that $\phi(0) = \nabla f(\mathbf{x}_0)$ and $\phi(1) = \nabla f(\mathbf{x}_\bigstar)$. An application of the chain rule yields

$$\phi'(t) = H_f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right)\left(\mathbf{x}_\bigstar - \mathbf{x}_0\right),$$

which is continuously differentiable on $[0,1]$ since f is continuously twice-differentiable on S. By the Fundamental Theorem of Calculus,

$$\nabla f(\mathbf{x}_\bigstar) - \nabla f(\mathbf{x}_0) = \int_{t=0}^{t=1} H_f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right)\left(\mathbf{x}_\bigstar - \mathbf{x}_0\right)\, dt. \tag{7.19}$$

If we substitute (7.19) into (7.18), we obtain

$$\begin{aligned}\mathbf{x}_1 - \mathbf{x}_\bigstar &= \mathbf{x}_0 - \mathbf{x}_\bigstar + H_f(\mathbf{x}_0)^{-1}\left(\nabla f(\mathbf{x}_\bigstar) - \nabla f(\mathbf{x}_0)\right) \\ &= H_f(\mathbf{x}_0)^{-1}\left(H_f(\mathbf{x}_0)\left(\mathbf{x}_0 - \mathbf{x}_\bigstar\right) + \left(\int_{t=0}^{t=1} H_f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right)\left(\mathbf{x}_\bigstar - \mathbf{x}_0\right)\, dt\right)\right). \\ &= H_f(\mathbf{x}_0)^{-1}\int_{t=0}^{t=1}\left(H_f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right) - H_f(\mathbf{x}_0)\right)\left(\mathbf{x}_\bigstar - \mathbf{x}_0\right)\, dt.\end{aligned} \tag{7.20}$$

We now apply the result from (7.20) to bound the norm of the vector difference, $\mathbf{x}_1 - \mathbf{x}_\bigstar$. In this process we utilize the fact that the eigenvalues of $H_f(\mathbf{x}_0)^{-1}$ are the reciprocals of the eigenvalues of $H_f(\mathbf{x}_0)$. Hence, $\left\|H_f(\mathbf{x}_0)^{-1}\right\| \leq m$. Using this bound, along with properties of the definite integral, and the Lipschitz continuity of $H_f(\mathbf{x})$, we obtain the following sequence of inequalities:

$$\begin{aligned}
\left\|\mathbf{x}_1 - \mathbf{x}_\bigstar\right\| &= \left\| H_f(\mathbf{x}_0)^{-1} \int_{t=0}^{t=1} \Big(H_f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right) - H_f(\mathbf{x}_0)\Big)\left(\mathbf{x}_\bigstar - \mathbf{x}_0\right)\, dt \right\| \\
&\leq \left\| H_f(\mathbf{x}_0)^{-1} \right\| \int_{t=0}^{t=1} \left\| H_f\left(t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0\right) - H_f(\mathbf{x}_0) \right\| \left\| \mathbf{x}_\bigstar - \mathbf{x}_0 \right\| \, dt \\
&\leq \left\| H_f(\mathbf{x}_0)^{-1} \right\| \int_{t=0}^{t=1} M \left\| t\mathbf{x}_\bigstar + (1-t)\mathbf{x}_0 - \mathbf{x}_0 \right\| \left\| \mathbf{x}_\bigstar - \mathbf{x}_0 \right\| \, dt \\
&= \left\| H_f(\mathbf{x}_0)^{-1} \right\| \int_{t=0}^{t=1} M \left\| \mathbf{x}_\bigstar - \mathbf{x}_0 \right\|^2 t \, dt \\
&= \frac{M}{2m} \left\| \mathbf{x}_\bigstar - \mathbf{x}_0 \right\|^2 .
\end{aligned}$$

Hence,

$$\left\|\mathbf{x}_1 - \mathbf{x}_\bigstar\right\| \leq \frac{M}{2m} \left\| \mathbf{x}_0 - \mathbf{x}_\bigstar \right\|^2 .$$

Since $\|\mathbf{x}_0 - \mathbf{x}_\bigstar\| < \sqrt{\dfrac{2m}{M}} \cdot \delta$, we see that $\left\|\mathbf{x}_1 - \mathbf{x}_\bigstar\right\| \leq \delta^2 < \delta$, so the preceding argument can be applied again, this time starting at $\mathbf{x}_1$. Repeating the results, we eventually obtain inequality (7.16) with $C = \dfrac{M}{2m}$, which completes the proof. □

Because inequality (7.16) implies that

$$\frac{\left\|\mathbf{x}_{k+1} - \mathbf{x}_\bigstar\right\|}{\left\|\mathbf{x}_k - \mathbf{x}_\bigstar\right\|^2} \leq C,$$

Theorem 7.2.1 can be rephrased as saying the sequence of iterates, starting at $\mathbf{x}_0$, converges to $\mathbf{x}_\bigstar$ at a *quadratic rate*. Because the denominator in the ratio is squared, this type of convergence is more rapid than linear convergence, as described following the statement of Theorem 7.1.2.

Unfortunately, while Theorem 7.2.1 guarantees quadratic convergence provided $\mathbf{x}_0$ is chosen sufficiently close $\mathbf{x}_\bigstar$, the proof given here does not explicitly indicate "how close" is good enough. Part of the difficulty stems from the facts we must establish that $H_f(\mathbf{x})$ is Lipschitz continuous and also determine the value of the Lipschitz constant, M. In most cases, Lipschitz continuity itself follows from "smoothness" conditions on f. For example, if all third-order partial derivatives of f are continuous in a neighborhood of $\mathbf{x}_\bigstar$, then Lipschitz continuity holds in that neighborhood. This follows from a higher-dimensional version of the Mean Value Theorem. Bounds on the third-order partial derivatives can then be used to estimate M. In practice, however, this

work is not necessary as convergence of the iterates can occur even if $\mathbf{x}_0$ is relatively far from $\mathbf{x}_\bigstar$.

In the general case of a twice-differentiable function, Newton's Method and the Steepest Descent Method frequently complement one another. The former method converges more rapidly when iterates are near the global minimum and less rapidly when they are far away. For the Steepest Descent method, the opposite is true.

Minimization of the *ConPro* objective function,
$f(x_1, x_2) = -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2$, illustrates this principle. Table 7.5 lists the first five iterations of both methods using an initial value of $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$.

TABLE 7.5: Results of the Steepest Descent and Newton's Methods applied to the *ConPro* objective function

	Steepest Descent Method		Newton's Method	
k	$\mathbf{x}_k^t$	$\|\mathbf{x}_k - \mathbf{x}_\bigstar\|$	$\mathbf{x}_k^t$	$\|\mathbf{x}_k - \mathbf{x}_\bigstar\|$
0	[10, 10]	119.76	[10, 10]	119.76
1	[91.680, 85.907]	16.373	[28.063, 29.111]	93.518
2	[82.493, 95.792]	7.444	[56.819, 61.720]	50.103
3	[87.236, 100.200]	1.435	[80.0760, 91.040]	12.714
4	[86.558, 100.930]	0.892	[86.7370, 100.922]	0.801
5	[87.125, 101.455]	0.175	[87.110, 101.626]	0.004

Both methods demonstrate convergence of iterates to the global minimum, but they do so in a different manner. For small k, the error $\|\mathbf{x}_k - \mathbf{x}_\bigstar\|$ using the Steepest Descent Method is much less than obtained using Newton's Method. Starting at $k = 4$, this trend reverses. In fact, if $\epsilon = .01$, the Steepest Descent Method requires nine iterations before $\nabla f(\mathbf{x}_k) < \epsilon$, whereas, Newton's Method requires only five. That one of these methods is more effective when iterates are far from the global minimum and the other is more effective when iterates are near prompts the question of whether it is possible to construct a single procedure that blends Steepest Descent and Newton Methods so as to be effective in either case. In the next section, we see that this is the case and combine the best of both techniques, creating a new procedure known as the *Levenberg-Marquardt Algorithm*, which is particularly well-suited for approximating solutions of unconstrained NLPs, especially those arising in the area of nonlinear regression.

Exercises Section 7.2

1. Suppose $f : \mathbb{R}^2 \to \mathbb{R}$ is given by $f(x_1, x_2) = x_1^4 - 8x_1^3 + 24x_1^2 - 32x_1 + 4x_2^2 - 4x_2$. For what initial values, $\mathbf{x}_0$, will Newton's method fail to produce a sequence of iterates?

2. *Rosenbrock's valley function*, $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$, is useful for comparing the efficiency of various numeric optimization algorithms. Figure 7.6 illustrates the graph of this function, which varies significantly in magnitude in a neighborhood of the origin.

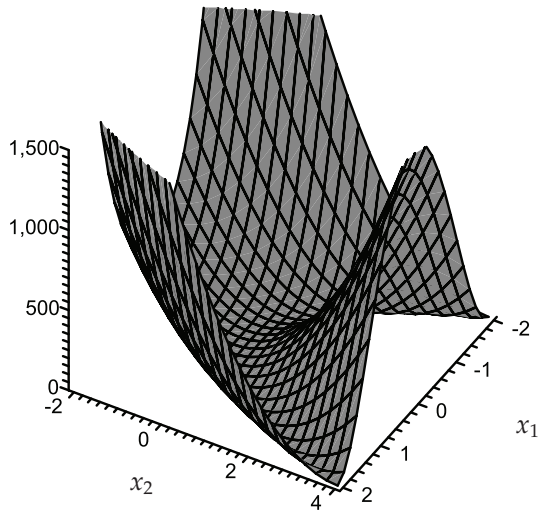


FIGURE 7.6: Rosenbrock's valley function.

(a) Verify that $\mathbf{x}_\star = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ is both a local and global minimum.

(b) Suppose $\mathbf{x}_0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$. Calculate the Newton direction of f at $\mathbf{x}_0$. Use your result to calculate the first Newton Method iterate, $\mathbf{x}_1$.

(c) How many iterations of Newton's Method are required to estimate the minimum of f with a tolerance of .01?

(d) Now determine the optimal descent direction of f at $\mathbf{x}_0$. Use your result to calculate the first Steepest Descent Method iterate, $\mathbf{x}_1$.

(e) Calculate two more Steepest Descent Method iterates, $\mathbf{x}_2$ and $\mathbf{x}_3$. Using the eigenvalues of $H_f(\mathbf{x}_\bigstar)$, verify that formula (7.9) from Section 7.1 is satisfied for these iteration values.

(f) How many iterations of the Steepest Descent Method are required to estimate the minimum of f with a tolerance of .01?

3. Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$ is defined by $f(x_1, x_2) = \ln(1 + x_1^2) + x_2^2$ and that $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$. Recall that the Steepest Descent Method, starting with $\mathbf{x}_0$, produces a sequence of iterates converging to the global minimum of f at the origin. Show that Newton's Method fails to produce such a convergent sequence. Then experiment with other initial values, $\mathbf{x}_0$, and try to account for the varying outcomes by using the Hessian of f.

4. Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$ is defined by $f(x_1, x_2) = x_1^4 + x_2^2$.

(a) Verify that f is convex on $\mathbb{R}^2$ and that, starting from any $\mathbf{x}_0$, in $\mathbb{R}^2$, the sequence of Newton Method iterates converges to the global minimum $\mathbf{x}_\bigstar = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

(b) Show explicitly that convergence is not quadratic, and explain why the hypothesis of Theorem 7.2.1 is not satisfied. (Hint: To construct a sequence that fails to exhibit quadratic convergence, consider setting $x_2 = 0$. Then Newton's Method yields a sequence of points on the x_1-axis that converge to zero, but not at a quadratic rate.)

7.3 The Levenberg-Marquardt Algorithm

7.3.1 Interpolating between the Steepest Descent and Newton Methods

The *Levenberg-Marquardt Algorithm* combines the Steepest Descent and Newton Methods in a manner that emphasizes the first of the two when $\mathbf{x}_k$ is far from a local minimum of f and the second when $\mathbf{x}_k$ is close. The method was first developed in 1944 by K. Levenberg and later refined in the early 1960s by D. W. Marquardt [23],[28].

Suppose $S \subseteq \mathbb{R}^n$ is open, $f : S \to \mathbb{R}$ is twice-differentiable, and that we wish to solve the unconstrained NLP,

$$\text{minimize } f(\mathbf{x}), \qquad \mathbf{x} \in S.$$

Assume $\mathbf{x}_0$ belongs to S. The *Levenberg Method* starts with the initial value $\mathbf{x}_0$ and, at each iteration, uses a *trial iterate* formula given by

$$\mathbf{w} = \mathbf{x}_k - \left[H_f(\mathbf{x}_k) + \lambda I_n\right]^{-1} \nabla f(\mathbf{x}_k) \qquad k = 0, 1, 2, \ldots. \tag{7.21}$$

In this formula, I_n denotes the n-by-n identity matrix, λ is a positive *damping parameter*, whose role is best understood by comparing extreme cases. If $\lambda = 0$, then (7.21) reduces to the iteration formula for Newton's Method. If λ is extremely large, then

$$\mathbf{w} \approx \mathbf{x}_k - \lambda^{-1} \nabla f(\mathbf{x}_k),$$

so that $\mathbf{w}$ is essentially the first Steepest Descent Method iterate of f starting from $\mathbf{x}_k$. Outside of these two extreme cases, λ controls the extent to which each method contributes to the interpolation of the two. Of course, we also assume λ yields a matrix sum, $H_f(\mathbf{x}_k) + \lambda I_n$, that is invertible. This will certainly be the case if λ is sufficiently large, as then $H_f(\mathbf{x}_k) + \lambda I_n \approx \lambda I_n$.

7.3.2 The Levenberg Method

The key to using (7.21) effectively is to adjust the damping parameter if the trial iterate has an objective value no less than that of $\mathbf{x}_k$. We start with a relatively small value of λ so that our interpolation formula is biased toward Newton's Method and hence, in light of Theorem 7.2.1, is more likely to yield a sequence converging to a minimum at a quadratic rate. If our choice of λ results in a value of $\mathbf{w}$, whose objective value is less than that of $\mathbf{x}_k$, i.e., if $f(\mathbf{w}) < f(\mathbf{x}_k)$, then we retain the trial iterate and set $\mathbf{x}_{k+1} = \mathbf{w}$. From there, we decrease the damping parameter, λ, and calculate a new trial iterate, $\mathbf{w}$, where $\mathbf{x}_{k+1}$ replaces $\mathbf{x}_k$ in (7.21). On the other hand, if $f(\mathbf{w}) \geq f(\mathbf{x}_k)$, then we increase λ until we obtain a new trial iterate, $\mathbf{w}$, for which $f(\mathbf{w}) < f(\mathbf{x}_k)$. This

process is repeated as necessary and leads to the method of Levenberg. We now summarize the steps of the algorithm.

The Levenberg Method

To obtain an approximate solution of NLP (7.1) under the assumption f is twice-differentiable on S:

1. Start with an initial value $\mathbf{x}_0$ and a small, positive damping parameter λ.
2. For $k \geq 0$, calculate the trial iterate,
$$\mathbf{w} = \mathbf{x}_k - \left[H_f(\mathbf{x}_k) + \lambda I_n\right]^{-1} \nabla f(\mathbf{x}_k). \tag{7.22}$$
3. If $f(\mathbf{w}) < f(\mathbf{x}_k)$, then set $\mathbf{x}_{k+1} = \mathbf{w}$, decrease λ, and return to (2) with $\mathbf{x}_k$ replaced by $\mathbf{x}_{k+1}$.
4. If $f(\mathbf{w}) \geq f(\mathbf{x}_k)$, then increase λ and return to (2).
5. Repeat the process until a desired level of tolerance has been achieved, e.g., $\left\|\nabla f(\mathbf{x}_k)\right\|$ is smaller than a specified value ϵ.

The algorithm does not stipulate the extent to which we should decrease or increase λ in (3) and (4), respectively. We return to this issue shortly. For now, mere experimentation with various positive values will suffice.

As an example, we now apply one iteration to the unconstrained *ConPro Manufacturing Company* NLP,

$$\text{minimize } f(x_1, x_2) = -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2,$$
$$\text{where } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in S = \mathbb{R}^2_+ = \{(x_1, x_2) \mid x_1, x_2 > 0\}, \tag{7.23}$$

with initial value $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$ and $\lambda = .01$. Calculations establish that $f(\mathbf{x}_0) = -4038.1$, $\nabla f(\mathbf{x}_0) = \begin{bmatrix} -126.904 \\ -117.93 \end{bmatrix}$, and $H_f(\mathbf{x}_0) = \begin{bmatrix} 23.845 & -15.897 \\ -15.897 & 21.196 \end{bmatrix}$. When these values are substituted into the interpolation formula (7.22), we obtain $\mathbf{w} = \begin{bmatrix} 23.470 \\ 24.242 \end{bmatrix}$. Since $f(\mathbf{w}) = -7097.364 < f(\mathbf{x}_0)$, we set $\mathbf{x}_1 = \mathbf{w}$, decrease λ, and repeat the process with $\mathbf{x}_0$ replaced by $\mathbf{x}_1$.

Waypoint 7.3.1. Perform the second iteration of the Levenberg Method for the unconstrained *ConPro Manufacturing Company* NLP, using the previously obtained value of $\mathbf{x}_1$, together with $\lambda = .005$.

7.3.3 The Levenberg-Marquardt Algorithm

For large values of λ, the Levenberg Method iterates depend primarily upon the gradient ∇f. We can incorporate useful second derivative information without introducing an excessive amount of extra work, insofar as matrix inversion is concerned, if we replace I_n in (7.22) with the diagonal matrix

$$\text{Diag}\left(H_f(\mathbf{x}_k)\right) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}\Big|_{\mathbf{x}=\mathbf{x}_k} & 0 & \dots & 0 \\ 0 & \frac{\partial^2 f}{\partial x_2^2}\Big|_{\mathbf{x}=\mathbf{x}_k} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial^2 f}{\partial x_n^2}\Big|_{\mathbf{x}=\mathbf{x}_k} \end{bmatrix}. \tag{7.24}$$

Note that $\text{Diag}\left(H_f(\mathbf{x}_k)\right)$ utilizes only those entries along the diagonal of $H_f(\mathbf{x}_k)$.

Substitution of (7.24) for I_n in (7.22) leads to a new interpolation formula for computing the trial iterate:

$$\mathbf{w} = \mathbf{x}_k - \left[H_f(\mathbf{x}_k) + \lambda \text{Diag}\left(H_f(\mathbf{x}_k)\right)\right]^{-1} \nabla f(\mathbf{x}_k). \tag{7.25}$$

We now introduce a decision process for accepting the trial iterate, one that is similar to that used for the Levenberg Method, but that depends upon both an *initial damping parameter*, λ_0, together with a *scaling factor*, $\rho > 1$. At each iteration, we compute the trial iterate using the previous iterate, $\mathbf{x}_k$, along with a value of λ that depends upon λ_0, ρ, and k. For many applications, $\lambda_0 = .001$ and $\rho = 10$ will suffice [7]. This new decision process leads to the Levenberg-Marquardt Algorithm.

The Levenberg-Marquardt Algorithm

To obtain an approximate solution of NLP (7.1) under the assumption f is twice-differentiable on S:

1. Start with an initial value, $\mathbf{x}_0$, in S, an initial damping parameter, λ_0, and a scaling parameter, ρ. For $k \geq 0$ do the following:
2. Determine a trial iterate, $\mathbf{w}$, using (7.25) with $\lambda = \lambda_k$.
3. If $f(\mathbf{w}) < f(\mathbf{x}_k)$, where $\mathbf{w}$ is determined in (2), then set $\mathbf{x}_{k+1} = \mathbf{w}$ and $\lambda_{k+1} = \lambda_k \rho^{-1}$. Return to (2), replace k with $k+1$, and compute a new trial iterate.
4. If $f(\mathbf{w}) \geq f(\mathbf{x}_k)$ in (3), determine a new trial iterate, $\mathbf{w}$, using (7.25) with $\lambda = \lambda_k$.
5. If $f(\mathbf{w}) < f(\mathbf{x}_k)$, where $\mathbf{w}$ is determined in (4), then set $\mathbf{x}_{k+1} = \mathbf{w}$ and $\lambda_{k+1} = \lambda_k$. Return to (2), replace k with $k+1$, and compute a new trial iterate.

6. If $f(\mathbf{w}) \geq f(\mathbf{x}_k)$ in (5), then determine the smallest value of m so that when a trial iterate, $\mathbf{w}$, is computed using (7.25) with $\lambda = \lambda_k \rho^m$, then $f(\mathbf{w}) < f(\mathbf{x}_k)$. Set $\mathbf{x}_{k+1} = \mathbf{w}$ and $\lambda_{k+1} = \lambda_k \rho^m$. Return to (2), replace k with $k + 1$, and compute a new trial iterate.

7. Terminate the algorithm when $\left\|\nabla f(\mathbf{x}_k)\right\| < \epsilon$, where ϵ is a specified tolerance.

An important observation to make is that in step (3), by setting $\lambda_{k+1} = \lambda_k \rho^{-1}$, we are emphasizing to a greater extent the Newton Method component in the interpolation formula (7.25), with the hope that the sequence of iterates converges to the minimum at a quadratic rate. If our decision-making process takes us to (6), then setting $\lambda_{k+1} = \lambda_k \rho^m > \lambda_k$ signifies we are forced to emphasize the Steepest Descent Method component in (7.25).

An application of the Levenberg-Marquardt Algorithm to the unconstrained *ConPro Manufacturing Company* NLP (7.23) with $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $\lambda_0 = .001$, $\rho = 10$, and tolerance $\epsilon = .01$ yields results shown in Table 7.6.

TABLE 7.6: Results of Levenberg-Marquardt Algorithm applied to the unconstrained *ConPro Manufacturing Company* NLP with a tolerance of $\epsilon = .01$, $\lambda = .0001$, and $\rho = 10$

k	$\mathbf{x}_k^t$	$\left\|\mathbf{x}_k - \mathbf{x}_\bigstar\right\|$
0	[10, 10]	119.759
1	[28.057, 29.105]	93.527
2	[56.811, 67.711]	51.116396
3	[80.072, 91.034]	12.721
4	[86.736, 100.920]	.802
5	[87.111, 101.626]	0.004

Table 7.2 indicates that eight iterations of the Steepest Descent Method are required to achieve the same level of tolerance. Newton's Method only requires five. Thus, for this example the Levenberg-Marquardt Algorithm requires fewer iterations than the Steepest Descent Method and a number equal to that of Newton's Method. It should not come as a surprise then that, under appropriate conditions, this new algorithm exhibits quadratic convergence [47].

7.3.4 A Maple Implementation of the Levenberg-Marquardt Algorithm

The `LevenbergMarquardt` procedure found in the worksheet **Levenberg-Marquardt Algorithm.mw** functions in a manner similar to the

`SteepestDescentMethod` and `NewtonsMethod` procedures from Sections 7.1 and 7.2, respectively. However, we must take into account different choices of the damping parameter and scaling factor. Our procedure call takes the form

`LevenbergMarquardt(`*`function`*`,`*`initial`*`,`*`N`*`,`*`tolerance`*`, `*`damp`*`,`*`scale`*`),`

where *`function`* is a twice-differentiable expression to be minimized, *`initial`* is the initial value, $\mathbf{x}_0$, (written as a list), *`N`* is the maximum number of iterations, *`tolerance`* is the specified tolerance, *`damp`* is the initial damping parameter, and *`scale`* is the scaling factor. The algorithm terminates when either the number of iterations is reached or the gradient norm is less than the desired tolerance. The returned value is the approximate minimum value. Intermediate iterates leading to the approximate minimum are stored in a list of points, `iterates`, a global variable accessible outside of the procedure. In the procedure, we utilize an auxiliary function, *d*, that returns formula (7.25) in terms of an input value, λ, at each iteration.

The following worksheet constructs the procedure and approximates the minimum of the *ConPro* objective, $f(x_1, x_2) = -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2$ using $\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $N = 5$, a tolerance of $\epsilon = .01$, a damping parameter of .001, and a scaling factor of 10.

```
> restart: with(VectorCalculus):with(LinearAlgebra):
> LevenbergMarquardt := proc(function, initial, N, tolerance, damp,
  scale)
  local x,f,d,Delf,Hf, DiagH,lambda,rho,w,j,m,epsilon;
  global iterates:
  # Create local and global variables.
  x:=array(0 .. N): # Define array of iterates.
  f:=unapply(function, [x1, x2]):
  # Create function.
  Delf:=unapply(Gradient(function, [x1, x2]), [x1, x2]):
  # Create corresponding gradient function.
  Hf:=unapply(Hessian(function, [x1, x2]), [x1, x2]):
  # Create corresponding Hessian matrix function.
  x[0]:=evalf(initial):
  # Set initial array value.
  epsilon := 10:
  # Set initial epsilon to a large value.
  lambda[0]:=damp: rho:=scale:
  # Set initial damping parameter and scaling factor.
  j:=0: while (tolerance <= epsilon and j<=N-1) do
  # Create loop structure to perform algorithm.
  DiagH := DiagonalMatrix(Diagonal(Hf(x[j][1], x[j][2]))):
  # Create diagonal matrix using Hessian entries.
```

```
d :=r-> convert(-MatrixInverse(Hf(x[j][1], x[j][2])
+r*DiagH).Delf(x[j][1], x[j][2]), list)
# Formulate auxiliary function.
w := x[j]+d(lambda[j]/rho);
# Create trial solution.
Test for acceptance of first trial solution.
if f(w[1], w[2]) < f(x[j][1], x[j][2]) then x[j+1] := w:
lambda[j+1] := lambda[j]/rho
else w := x[j]+d(lambda[j]);
# Test for acceptance of second trial solution.
if f(w[1], w[2]) < f(x[j][1], x[j][2]) then x[j+1] := w:
lambda[j+1]:=lambda[j]
else w := x[j];
# If necessary, scale damping factor
until a suitable trial solution found.
m:=1:while f(x[j][1], x[j][2]) <= f(w[1], w[2]) do
w:=x[j]+d(lambda[j]*rhoˆm):
m:= m+1 end do; x[j+1] := w: lambda[j+1]:=lambda[j]*rhoˆm
end if: end if;
epsilon := evalf(Norm(Delf(x[j+1][1], x[j+1][2]), Euclidean));
# Update epsilon using gradient.
j:=j+1:
# Increase loop index.
end do:
iterates:=[seq(x[i],i=0..j)]:RETURN(x[j]):end:
# Construct iterates and return last array value.
```

```
> f:=(x1,x2)->-1400x1ˆ(1/2)x2ˆ(1/3)+350x1+200x2;
# Enter function
```

$$f := (x_1, x_2) \rightarrow -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2$$

```
> LevenbergMarquardt(f(x1, x2), [10,10],5,.01,.001,10);
# Procedure call using initial value [10,10], bound of 5 iterations,
tolerance, .00, damping parameter of .001, and scaling factor
of 10.
```

$$[87.10975285, 101.6261946]$$

```
> iterates;
# Print iterates.
```

$$[[10., 10.], [28.00148014, 29.04621145], [56.73574951, 61.62388852],$$
$$[80.03451430, 90.98472144], [86.73264262, 100.9143105],$$
$$[87.10972680, 101.6261320]]$$

7.3.5 Nonlinear Regression

Levenberg's original intent was to develop an algorithm well-suited for solving nonlinear least-squares problems. Specifically, given a set of data pairs, $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$, where each y_k is real and each $\mathbf{x}_k$ belongs to $\mathbb{R}^n$, we wish to determine a function, f, of a certain type that minimizes the *sum of squared-errors*,

$$\sum_{k=1}^{N} \left(f(\mathbf{x}_k) - y_k\right)^2 . \tag{7.26}$$

Generally, we assume that f belongs to a family of functions that are related by one or more parameters. Multiple linear regression, as discussed in Section 6.3.5, is a particular example.

Unlike the case of multiple linear regression, where (7.26) results in an unconstrained NLP whose critical points are determined analytically, minimization of (7.26) in the nonlinear setting frequently requires a numeric algorithm.

Such was the case for the *ConPro Manufacturing Company* when it constructed its production function, which was based upon the data shown in Table 7.7.

TABLE 7.7: Production data for *ConPro Manufacturing Company*

x_1	x_2	$P(x_1, x_2)$
60	60	29
60	80	36
60	100	34
60	120	37
80	60	36
80	80	38
80	100	41
80	120	47
100	60	35
100	80	45
100	100	47
100	120	47
120	60	46
120	80	48
120	100	47
120	120	54

Recall x_1 and x_2 denote the number of units of material and labor, respectively, used to produce $P(x_1, x_2)$ units of pipe.

A traditional choice from economic theory for modeling production is the

Cobb-Douglas Production Function,

$$P(x_1, x_2) = x_1^{\alpha} x_2^{\beta}, \text{ where } \alpha, \beta > 0. \tag{7.27}$$

To determine the Cobb-Douglas function that best fits the data in Table 7.7, we minimize the *sum of squared-errors*

$$f(\alpha, \beta) = \sum_{k=1}^{16} \left(x_1^{\alpha} x_2^{\beta} - P(x_1, x_2) \right)^2, \qquad \alpha, \beta > 0, \tag{7.28}$$

where the sum is over the sixteen ordered triples $(x_1, x_2, P(x_1, x_2))$ in Table 7.7. If we apply the Levenberg-Marquardt Algorithm, using an initial value $\mathbf{x}_0 = \begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix} = \begin{bmatrix} .5 \\ .5 \end{bmatrix}$, initial damping factor $\lambda_0 = .001$, a scaling factor of $\rho = 10$, and a tolerance of $\epsilon = .001$, we obtain after five iterates, $\alpha \approx .510$ and $\beta \approx .322$. These values are quite close to those used by *ConPro* in its actual production formula.

Figure 7.7 illustrates a plot of the data triples (x_1, x_2, P) from Table 7.7 along with the "best-fit" production function $P(x_1, x_2) = x_1^{.510} x_2^{.320}$.

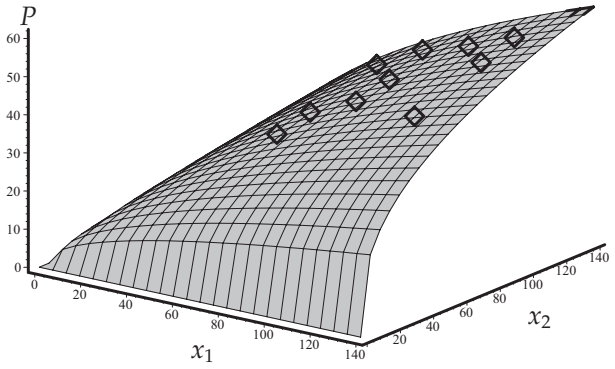


FIGURE 7.7: Plot of *ConPro Manufacturing Company* production data, together with best-fit Cobb-Douglass function $P(x_1, x_2) = x_1^{.510} x_2^{.320}$.

An important quantity associated with any regression problem is the *coefficient of determination*. This quantity lies between 0 and 1 and measures the proportion of variation of the data accounted for by the best-fit function. If the data points used for the regression are given by $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$ and if $\overline{y}$ denotes the mean of $\{y_1, y_2, \ldots, y_m\}$, this proportion then becomes

$$\frac{\sum_{k=1}^{m} \left(f(\mathbf{x}_k) - y_k \right)^2}{\sum_{k=1}^{m} \left(\overline{y} - y_k \right)^2}. \tag{7.29}$$

Direct calculation using (7.29) shows that the production function $P(x_1, x_2) = x_1^{.510} x_2^{.320}$ fits the data in Table 7.7 with a coefficient of determination equal to .94, thus indicating a strong goodness of fit.

7.3.6 Maple's `Global Optimization Toolbox`

The Steepest Descent, Newton, and Levenberg-Marquardt Methods are only three of many numeric algorithms used to approximate solutions of unconstrained NLPs. Their use is limited, however, in that they offer no guarantee of locating global solutions, local solutions, or even critical points for that matter.

There is a great deal of ongoing research in the field of global optimization, and several techniques, which are beyond the scope of this text, have been developed for solving unconstrained NLPs.

One specific tool developed during the last decade is known as *Lipschitz Global Optimization*, or LGO [39]. Implementation of LGO for various software programs, e.g., Maple and Excel, is now available as an add-on package. In Maple, for example, the `Global Optimization Toolbox` package is loaded using the command `with(GlobalOptimization)`, and the global minimum is computed using the command `GlobalSolve`, whose arguments essentially take the same form as those used for the `LPSolve` and `NLPSolve` commands.

A brief review of other various global optimization software packages may be found in [40]. Three texts devoted to the general topic of global optimization include [13], [16], [35], and [39].

Exercises Section 7.3

1. Recall Rosenbrock's banana function from Exercise 2 of Section 7.2, $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$.

   (a) Compute the first Levenberg Algorithm iterate, $\mathbf{x}_1$ using $\mathbf{x}_0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$, $N = 5$, and each of the following three damping factors:

   i. $\lambda = .1$
   ii. $\lambda = .01$
   iii. $\lambda = .001$

   Account for any differences you notice.

   (b) Compute the first four Levenberg-Marquardt Algorithm iterates

using $\mathbf{x}_0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$, $\lambda_0 = .001$, and $\rho = 10$. At each iteration, record the value of λ_k.

(c) How many iterations of the Levenberg-Marquardt Algorithm are necessary before $\left\|\nabla f\left(\mathbf{x}_k\right)\right\| < .01$? How does this result compare with the methods of Steepest Descent and Newton?

2. A logistic function frequently models the growth of an organism, which is initially exponential in nature but is eventually slowed due to one or more environmental factors. Such growth might be mass, diameter, height, etc. The general form of such a function can be written as

$$P(t) = \frac{\alpha}{\beta + e^{-kt}}. \tag{7.30}$$

where α, β, and k are parameters, with $k > 0$. A classic example illustrating how this nonlinear function models real world data is provided by the sunflower plant. Table 7.8, for example, displays data showing sunflower height measured weekly over a period of 12 weeks.

TABLE 7.8: Sunflower growth data

t (days)	H (centimeters)
0	0.00
7	17.93
14	36.36
21	67.76
28	98.10
35	131.00
42	169.50
49	205.50
56	228.30
63	247.10
70	250.50
77	253.80
84	254.50

(a) Find the logistic function P of the form given in Equation 7.30 that best fits the data provided in Table 7.8 in the sense of least squares. That is, determine α, β, and k using Table 7.8 and nonlinear regression. (Hint: To determine sensible initial values of α, β, and k, you may wish to graph the data points first, using Maple's `pointplot` command. Then experiment with various values of α, β, and k. For each choice, use Maple to construct the graph of f from (7.30). Superimpose this graph on that of the data points and see if your values of α, β, and k make reasonably good initial choices.)

(b) What is the coefficient of determination associated with the best-fit function you just obtained?

3. In chemical kinetics, if an unstable chemical species spontaneously decays with constant decay probability per unit time interval, its concentration C is governed by exponential decay and thus represented as

$$C(t) = C_0 e^{-kt},$$

where C_0 denotes the initial concentration and k is a constant.[1] The average lifetime of the species is then $\tau = \frac{1}{k}$. One goal of some chemical kinetics experiments is to determine the value of τ (or, equivalently, of k.) Frequently the concentration C itself is not experimentally observable, but some function $Q = aC + b$, where a and b are real but unknown constants, is observable instead. A typical data set therefore consists of a set of observations of Q at various times t, represented as a sequence $\{(t_1, Q_1), (t_2, Q_2), \ldots, (t_m, Q_m)\}$.

(a) Show that

$$Q(t) = Ae^{-kt} + B, \tag{7.31}$$

where A and B are constants related to a, b, and C_0.

In one experiment, ions in an excited electronic state were suddenly produced by a pulsed laser. These ions then decayed by emission of orange light. Since the intensity of the light is proportional to the concentration of the excited species, we may choose Q as intensity in (7.31). A photomultiplier converted this intensity to a signal recorded by an oscilloscope as a voltage. The result of 25 consecutive signal intensity samples is shown in Table 7.9.

(b) Find the function (7.31) that best fits the data in the sense of least-squares. That is, determine A, B, and k using Table 7.9 and nonlinear regression. Once you have determined A, B, and k, compute, τ, the average lifetime of the ion, in microseconds.

(c) What is the coefficient of determination associated with the best-fit function you just obtained?

[1] Data and background information provided by Professor George McBane, Department of Chemistry, Grand Valley State University.

TABLE 7.9: Time-intensity data for pulsed-laser experiment

t (microseconds)	Q (millivolts)
0.075	1.90757
0.135	1.40464
0.195	1.31003
0.255	1.14646
0.315	0.967627
0.375	0.785132
0.435	0.702124
0.495	0.605078
0.555	0.516577
0.615	0.387793
0.675	0.443945
0.735	0.337744
0.795	0.259009
0.855	0.188818
0.915	0.204077
0.975	0.150977
1.035	0.0588135
1.095	0.0472168
1.155	0.0520996
1.215	0.0429443
1.275	0.0130371
1.335	0.00327148
1.395	-0.0449463
1.455	-0.0296875
1.515	-0.0333496

Chapter 8

Methods for Constrained Nonlinear Problems

8.1 The Lagrangian Function and Lagrange Multipliers

Having focused exclusively on unconstrained NLPs in Chapters 6 and 7, we now turn our attention to the constrained NLP, whose general form is given by

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to} \\ &\quad g_1(\mathbf{x}) \leq 0 \\ &\quad g_2(\mathbf{x}) \leq 0 \\ &\quad \vdots \\ &\quad g_m(\mathbf{x}) \leq 0 \\ &\quad h_1(\mathbf{x}) = 0 \\ &\quad h_2(\mathbf{x}) = 0 \\ &\quad \vdots \\ &\quad h_p(\mathbf{x}) = 0. \end{aligned} \tag{8.1}$$

We assume that $S \subseteq \mathbb{R}^n$ is convex and that all functions are differentiable mappings from S to $\mathbb{R}$. A constraint involving a function g_i, where $1 \leq i \leq m$, is of *inequality-type*, whereas that involving h_j, where $1 \leq j \leq p$, is of *equality-type*. In the spirit of linear programming terminology, we say that $\mathbf{x} \in S$ is a *feasible solution* of (8.1) if it satisfies all the constraints.

Our prototype problem is the constrained *ConPro Manufacturing Company* NLP introduced in Section 6.1.3. If x_1 and x_2 denote the number of units of material and labor, respectively, then the NLP, stated as a minimization problem, is given by

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2 \\ \text{subject to} & \\ & g_1(\mathbf{x}) = 350x_1 + 200x_2 - 40000 \leq 0 \\ & g_2(\mathbf{x}) = x_1 - 3x_2 \leq 0 \\ & g_3(\mathbf{x}) = -x_1 \leq 0 \\ & g_4(\mathbf{x}) = -x_2 \leq 0. \end{aligned} \tag{8.2}$$

We used the objective function to solve the unconstrained *ConPro* NLP in Chapters 6 and 7. Constraints reflect the limitation on funds available for purchasing material and labor, the requirement of adequate labor to produce pipes from acquired material, and sign restrictions on x_1 and x_2.

8.1.1 Some Convenient Notation

Let $\mathbf{g} : S \to \mathbb{R}^m$ and $\mathbf{h} : S \to \mathbb{R}^p$ be vector-valued functions defined by $\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_m(\mathbf{x}) \end{bmatrix}$ and $\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_p(\mathbf{x}) \end{bmatrix}$. Then (8.1) can be written more compactly as

$$\begin{aligned} \text{minimize } & f(\mathbf{x}) \\ \text{subject to} & \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{aligned} \tag{8.3}$$

The function $\mathbf{g}$ is a vector-valued function whose components are of the type discussed in Section 6.2. Consequently, we define $\mathbf{g}$ to be differentiable at $\mathbf{x}_0$ in S if and only if each component function g_i, where $1 \leq i \leq m$, is differentiable there. When this occurs, we write

$$\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{x}_0) + \mathbf{J}_\mathbf{g}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \|\mathbf{x} - \mathbf{x}_0\|\mathbf{R}(\mathbf{x}, \mathbf{x}_0), \tag{8.4}$$

where all components of $\mathbf{R}(\mathbf{x}, \mathbf{x}_0) \in \mathbb{R}^m$ tend to $\mathbf{0}$ as $\mathbf{x} \to \mathbf{x}_0$ and where $\mathbf{J}_\mathbf{g}(\mathbf{x}_0)$ is an m-by-n matrix, called the *Jacobian* of $\mathbf{g}$ evaluated at $\mathbf{x}_0$. It is the matrix whose ith row, $1 \leq i \leq m$, is given by the gradient vector transpose, $\nabla g_i(\mathbf{x}_0)^t$.

The Jacobian matrix yields the linear approximation

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}(\mathbf{x}_0) + \mathbf{J}_\mathbf{g}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \text{ for } \mathbf{x} \approx \mathbf{x}_0. \tag{8.5}$$

Naturally, the preceding analysis can be modified when $\mathbf{g}$ is replaced by $\mathbf{h}$, in which case $\mathbf{J_h}(\mathbf{x}_0)$ is a p-by-n matrix.

In Maple, the Jacobian is computed using the `Jacobian` command, located in the `VectorCalculus` package. Its general form is given by

`Jacobian(`*`vector of expressions`*`,`*`variable list`*`),`

and it computes the Jacobian of *`vector of expressions`* with respect to the variables given in *`variable list`*. The output of the command is a matrix. Here is syntax illustrating how to compute the Jacobian of the function $\mathbf{g}$, where $\mathbf{g}$ is formed using the four constraints of the *ConPro Manufacturing Company* NLP (8.2). In this example, we construct the Jacobian as a function of the vector, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

```
> with(VectorCalculus):
> g1:=(x1,x2)->350*x1+200*x2-40000:
> g2:=(x1,x2)->x1-3*x2:
> g3:=(x1,x2)->-x1:
> g4:=(x1,x2)->-x2:
> g:=(x1,x2)-><g1(x1,x2),g2(x1,x2),g3(x1,x2),g4(x1,x2)>:g(x1,x2);
```

$$(350x_1 + 200x_2 - 40000)\mathbf{e}_{x_1} + (x_1 - 3x_2)\mathbf{e}_{x_2} - x_1\mathbf{e}_{x_3} - x_2\mathbf{e}_{x_4}$$

```
> Jg := unapply(Jacobian(g(x1, x2), [x1, x2]), [x1, x2]): Jg(x1,x2);
```

$$\begin{bmatrix} 350 & 200 \\ 1 & -3 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

8.1.2 The Karush-Kuhn-Tucker Theorem

Key to solving constrained NLPs is the *Lagrangian function*, a single function that incorporates the objective as well as all constraints.

Definition 8.1.1. The Lagrangian function corresponding to NLP (8.1) is the function $L : S \times \mathbb{R}^{m+p} \to \mathbb{R}$ defined by

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x}) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^t \mathbf{g}(\mathbf{x}) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}), \end{aligned} \tag{8.6}$$

where $\mathbf{x}$ belongs to $\mathbb{R}^n$ and where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are vectors in $\mathbb{R}^m$ and $\mathbb{R}^p$, respectively, denoted by $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix}$ and $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix}$.

Note that in each product, $\boldsymbol{\lambda}^t\mathbf{g}(\mathbf{x})$ and $\boldsymbol{\mu}^t\mathbf{h}(\mathbf{x})$, there exists a natural correspondence between each constraint of the NLP (8.1) and a component of $\boldsymbol{\lambda}$ or $\boldsymbol{\mu}$. In particular, a constraint of inequality-type (respectively, of equality-type) corresponds to a component of $\boldsymbol{\lambda}$ (respectively, to a component of $\boldsymbol{\mu}$).

Our subsequent investigation focuses on the Lagrangian function. Using this tool, we obtain important information regarding solutions of (8.1). The overall process closely resembles that from Sections 6.2-6.4 in that we first determine necessary conditions $\mathbf{x}_0$ in $\mathbb{R}^n$ to be a solution of (8.1). Then we establish sufficient conditions for $\mathbf{x}_0$ to be a solution of this NLP. These tools, which were developed in the mid 1900s by Fritz John, William Karush, Harold William Kuhn, and Albert William Tucker, form the basis of constrained optimization theory.

Throughout this discussion, unless stated otherwise, we use the notation $\nabla L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ to denote the gradient of L with respect to the components of $\mathbf{x}$. A useful formula that expresses $\nabla L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ in terms of the functions that define L is given by

$$\begin{aligned} \nabla L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= \nabla f(\mathbf{x}_0) + \sum_{i=1}^{m} \lambda_{0,i} \nabla g_j(\mathbf{x}_0) + \sum_{j=1}^{p} \mu_{0,j} \nabla h_j(\mathbf{x}_0) \\ &= \nabla f(\mathbf{x}) + \mathbf{J}_{\mathbf{g}}(\mathbf{x})^t \boldsymbol{\lambda} + \mathbf{J}_{\mathbf{h}}(\mathbf{x})^t \boldsymbol{\mu}. \end{aligned} \tag{8.7}$$

We now state an important necessary condition, in terms of the Lagrangian, for a feasible solution of (8.3) to be optimal.

Theorem 8.1.1. Consider the NLP (8.3) and assume that f and all component functions of $\mathbf{g}$ and $\mathbf{h}$ are continuously differentiable on S. Suppose that $\mathbf{x}_0$ is a solution of the NLP, and let I denote the set of indices corresponding to *binding* inequality constraints. In other words

$$I = \{i \mid 1 \leq i \leq m \text{ and } g_i(\mathbf{x}_0) = 0\}. \tag{8.8}$$

Let V be the set of gradient vectors corresponding to both equality constraints and binding inequality constraints, each of which is evaluated at $\mathbf{x}_0$. In other words,

$$V = \{\nabla g_i(\mathbf{x}_0) \mid i \in I\} \cup \left\{\nabla h_j(\mathbf{x}_0) \mid 1 \leq j \leq p\right\}.$$

If V forms a linearly independent set of vectors, a provision we refer to as the

the *regularity condition*, then there exist unique vectors, $\boldsymbol{\lambda}_0 \in \mathbb{R}^m$ and $\boldsymbol{\mu}_0 \in \mathbb{R}^p$, having the following three properties:

$$\boldsymbol{\lambda}_0 \geq \mathbf{0} \tag{8.9}$$

$$\nabla L(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) = \nabla f(\mathbf{x}_0) + \mathbf{J_g}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 + \mathbf{J_h}(\mathbf{x}_0)^t \boldsymbol{\mu}_0 = \mathbf{0} \tag{8.10}$$

$$\lambda_{0,i} g_i(\mathbf{x}_0) = 0 \quad \text{for } 1 \leq i \leq m. \tag{8.11}$$

Theorem 8.1.1 is often referred to as the Karush-Kuhn-Tucker Theorem (or Karush-Kuhn-Tucker Necessary Conditions Theorem). We omit its proof, which can be found in more advanced texts [2][4]. Instead, we take a closer look at its meaning.

1. Entries of the vectors $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$ are referred to as the *Lagrange multipliers* associated with the NLP (8.3), and the vectors $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$ themselves as the *multiplier vectors*.

2. Any feasible value $\mathbf{x}_0$ for which conditions (8.9)-(8.11) hold is called a *Karush-Kuhn-Tucker*, or *KKT-point* of NLP (8.3). If the regularity condition also holds at $\mathbf{x}_0$, we say the KKT point is regular.

3. The regularity condition cannot be overlooked. If we omit this requirement altogether, the multiplier vectors satisfying conditions (8.9)-(8.11) need not exist in the first place, or, if they do exist, they need not be unique. Exercises 2 and 3 establish these facts.

4. As a consequence of conditions (8.7), (8.10), and (8.11),

$$\begin{aligned} \mathbf{0} &= \nabla L(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \\ &= \nabla f(\mathbf{x}_0) + \mathbf{J_g}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 + \mathbf{J_h}(\mathbf{x}_0)^t \boldsymbol{\mu}_0 \\ &= \nabla f(\mathbf{x}_0) + \sum_{i=1}^{m} \lambda_{0,i} \nabla g_j(\mathbf{x}_0) + \sum_{j=1}^{p} \mu_{0,j} \nabla h_j(\mathbf{x}_0) \\ &= \nabla f(\mathbf{x}_0) + \sum_{i \in I} \lambda_{0,i} \nabla g_i(\mathbf{x}_0) + \sum_{j=1}^{p} \mu_{0,j} \nabla h_j(\mathbf{x}_0). \end{aligned} \tag{8.12}$$

   In other words, $\nabla f(\mathbf{x}_0)$ is a linear combination of the vectors in V, with weights determined by the Lagrange Multipliers.

5. An inequality constraint of the form $g_i(\mathbf{x}) \leq 0$ corresponds to the sign restriction $\lambda_{0,i} \geq 0$, whereas $h_j(\mathbf{x}) = 0$ corresponds to a multiplier $\mu_{0,j}$ that is *unrestricted in sign*. Results from Section 4.1, such as those summarized in Table 4.4, suggest an intriguing connection between the multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$, and duality.

6. This connection with duality is reinforced if we observe that condition (8.11) corresponds to the complementary slackness property, Theorem 4.1.4, from Section 4.1. Stated in simple terms, at an optimal solution of the NLP, $\mathbf{x}_0$, which satisfies the theorem's hypotheses, each multiplier, $\lambda_{0,i}$, is zero, or the corresponding constraint, $\mathbf{g}_i(\mathbf{x})$, is *binding*. As a result of this fact, in subsequent discussions we shall refer to (8.11) as the *complementary slackness conditions*. Since $\lambda_{0,i} \geq 0$ and $g_i(\mathbf{x}_0) \leq 0$ for $1 \leq i \leq m$, complementary slackness can be expressed in matrix-vector form as $\boldsymbol{\lambda}_0\mathbf{g}(\mathbf{x}_0) = 0$.

To investigate these ideas in the context of a particular example and to understand how the solution of an NLP determines the Lagrange multiplier vectors, we consider the *ConPro Manufacturing Company* NLP (8.2). In the next section, we establish that the optimal solution is given by

$$\mathbf{x}_0 = \begin{bmatrix} 480/7 \\ 80 \end{bmatrix} \approx \begin{bmatrix} 68.57 \\ 80 \end{bmatrix}, \tag{8.13}$$

with corresponding objective value, $f(\mathbf{x}_0) = -9953.15$. Thus, using approximately 68.57 units of material and 80 units of labor, *ConPro Manufacturing Company* has a maximum profit of \$9953.15, given the constraints dictated in (8.2). Observe that this solution differs from that of the unconstrained version of the problem, which we solved in Section 6.4 and which had solution $\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix} \approx \begin{bmatrix} 87.11 \\ 101.63 \end{bmatrix}$ with corresponding profit approximately equal to \$10,162.96.

A simple calculation shows that at the optimal solution, only the first constraint, $g_1(\mathbf{x}) = 350x_1 + 200x_2 - 40000 \leq 0$ is binding. Thus, $\nabla g_1(x_0) = \begin{bmatrix} 350 \\ 200 \end{bmatrix}$ trivially satisfies the regularity condition, so by (8.11), $\lambda_{0,2} = \lambda_{0,3} = \lambda_{0,4} = 0$.

The Lagrangian has its gradient given by

$$\begin{aligned} \nabla L(\mathbf{x}, \boldsymbol{\lambda}) &= \nabla f(\mathbf{x}) + \mathbf{J}_{\mathbf{g}}(\mathbf{x})^t \boldsymbol{\lambda} \\ &= -\begin{bmatrix} \frac{700x_2^{\frac{1}{3}}}{\sqrt{x_1}} + 350 \\ \frac{1400\sqrt{x_1}}{3x_2^{\frac{2}{3}}} + 200 \end{bmatrix} + \begin{bmatrix} 350 & 1 & -1 & 0 \\ 200 & -3 & 0 & -1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} \\ &= \begin{bmatrix} -\frac{700x_2^{\frac{1}{3}}}{\sqrt{x_1}} + 350 + 350\lambda_1 + \lambda_2 - \lambda_3 \\ -\frac{1400\sqrt{x_1}}{3x_2^{\frac{2}{3}}} + 200 + 200\lambda_1 - 3\lambda_2 - \lambda_4 \end{bmatrix}. \end{aligned} \tag{8.14}$$

Evaluating ∇L at $\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix}$ and $\lambda_{0,2} = \lambda_{0,3} = \lambda_{0,4} = 0$ and setting each

component of the result to zero, we obtain

$$\lambda_{0,1} = \sqrt{\frac{7}{120}}\sqrt[3]{80} - 1 \approx .04. \tag{8.15}$$

These results demonstrate that at the optimal solution of NLP (8.2), $\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix}$, with the Lagrange multiplier vector given by $\boldsymbol{\lambda}_0 \approx \begin{bmatrix} .04 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. In light of the linear dependence relation spelled out by (8.12), we see that

$$\begin{aligned} \mathbf{0} &= \nabla f(\mathbf{x}_0) + \mathbf{J_g}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 \\ &= \nabla f(\mathbf{x}_0) + \lambda_{0,1} \nabla g_1(\mathbf{x}_0) \\ &\approx \nabla f(\mathbf{x}_0) + .04 \nabla g_1(\mathbf{x}_0). \end{aligned}$$

Thus, for the *ConPro Manufacturing Company* NLP, at the optimal solution, $\mathbf{x}_0$ the gradients of f and the first constraint function, g_1, are scalar multiples of one another, where the scaling factor is approximately $-.04$.

Figure 8.1 illustrates the feasible region for the *ConPro Manufacturing Company* NLP, the optimal solution, $\mathbf{x}_0 \approx \begin{bmatrix} 87.11 \\ 101.63 \end{bmatrix}$, $\nabla f(\mathbf{x}_0)$, and $\nabla g_1(\mathbf{x}_0)$. Vectors have been scaled for purposes of visualization.

8.1.3 Interpreting the Multiplier

The multiplier, $\lambda_{0,1} \approx .04$ in the *ConPro Manufacturing Company* NLP has a simple economic interpretation. First, observe that the units of the Lagrangian function in (8.14) are the same as the units of f, namely dollars (of profit). Since the units of g_1, the only binding constraint in the solution, are dollars (of available funds for material and labor), the units of $\lambda_{0,1}$ are dollar of profit per dollar of available material and labor. Put another way, the multiplier $\lambda_{0,1}$ is the instantaneous rate of change of the objective, f, with respect to the constraint function, g_1. This means that if g_1 changes by a small amount, small enough so that it remains a binding constraint in the new solution, we should notice the value of f at the optimal solution change by an amount approximately equal to the change in g_1, multiplied by $\lambda_{0,1}$.

For example, suppose the funds available to *ConPro* for material and labor increases by \$100 from \$40,000 to \$40,100. Then the new first constraint becomes $350x_1 + 200x_2 \leq 40,100$, or, equivalently, g_1 in (8.2) changes to

$$g_1(\mathbf{x}) = 350x_1 + 200x_2 - 40100. \tag{8.16}$$

In terms of Figure 8.1, such a change corresponds to moving slightly up and

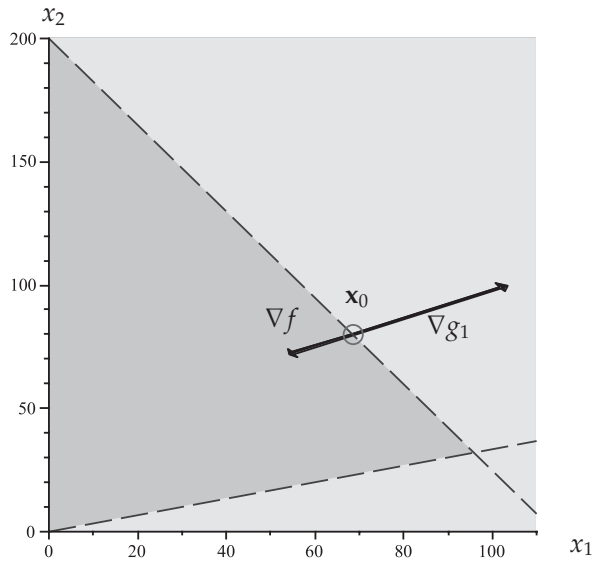


FIGURE 8.1: Feasible region of the *ConPro Manufacturing Company* NLP illustrating objective and constraint gradients at the solution, $\mathbf{x}_0$.

to the right the segment on the boundary of the feasible region that passes through $\mathbf{x}_0 = \begin{bmatrix} 784/9 \\ 2744/27 \end{bmatrix}$. The multiplier, $\lambda_{0,1} \approx .04$, indicates, that, under such a change, we should expect to see the objective value in the solution of the new NLP to change by $\Delta f \approx .04(-100) = -4$. This is a fairly good approximation, for if we replace the original first constraint by that in (8.16) and solve the resulting NLP (with Maple), we obtain a new optimal objective value of −9957.21. This value, less that of $f(\mathbf{x}_0) = -9953.15$, is approximately -4.06. Thus, with \$100 more funds available for capital and labor, *ConPro* 's profit increases by a mere \$4.06.

In essence, we can view the multiplier as playing a role in sensitivity analysis when we increase the bound on the right-hand side of a constraint. In fact, if we recall the concept of a *shadow price* introduced in Section 4.2.2, we see that the multiplier generalizes this idea to the nonlinear setting. What is more, shadow prices in the LP setting equalled the slack variable coefficients in the top row of the tableau, which in turn corresponded to the decision variables in the LP's dual formulation. That Lagrange multipliers are somehow connected to dual decision variables for an LP becomes the focus of further investigation in Exercise (5).

Exercises Section 8.1

1. Express each of the following NLPs in the forms (8.1) and (8.3). Use the provided solution to determine the corresponding Lagrange multipliers. Then sketch the feasible region along with the gradients of both f and the constraint functions evaluated at $\mathbf{x}_0$.

   (a)

$$\begin{aligned} &\text{minimize } f(x_1, x_2) = x_1^2 - 6x_1 + x_2^2 - 4x_2 \\ &\text{subject to} \\ &x_1 + x_2 \leq 3 \\ &x_1 \leq 1 \\ &x_2 \geq 0 \end{aligned}$$

   Solution: $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

   (b)

$$\begin{aligned} &\text{minimize } f(x_1, x_2) = 4x_1^2 - 12x_1 - x_2^2 - 6x_2 \\ &\text{subject to} \\ &x_1 + x_2^2 \leq 2 \\ &x_1 \leq 1 \\ &2x_1 + x_2 = 1 \end{aligned}$$

   Solution: $\mathbf{x}_0 = \begin{bmatrix} -1/4 \\ 3/2 \end{bmatrix}$

2. Consider the NLP given by

$$\begin{aligned} &\text{minimize } f(x_1, x_2) = x_1^2 - 4x_1 + x_2^2 \\ &\text{subject to} \\ &(1 - x_1)^3 - x_2 \geq 0 \\ &x_1, x_2 \geq 0, \end{aligned}$$

   whose solution is $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Including sign restrictions, this problem has three constraints.

(a) Verify that the regularity condition does not hold at $\mathbf{x}_0$ for this NLP.

(b) Show that there exists no vector $\boldsymbol{\lambda}_0$ in $\mathbb{R}^3$ for which $\boldsymbol{\lambda}_0 \geq \mathbf{0}$, $\nabla L(\mathbf{x}_0, \boldsymbol{\lambda}_0) = \mathbf{0}$, and the complementary slackness conditions hold. This result demonstrates that the regularity condition in the hypothesis of Theorem 8.1.1 is important for guaranteeing the existence of the multiplier vectors.

3. Consider the NLP given by

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1^3 - 3x_1 + x_2^2 - 6x_2 \\ \text{subject to} & \\ & x_1 + x_2 \leq 1 \\ & x_1 + x_2^2 + x_2 \leq 2 \\ & x_1, x_2 \geq 0, \end{aligned}$$

whose solution is $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Including sign restrictions, this problem has four constraints.

(a) Verify that the regularity condition does not hold at $\mathbf{x}_0$ for this NLP.

(b) Construct at least two different multiplier vectors, $\boldsymbol{\lambda}_0$, in $\mathbb{R}^4$ for which $\boldsymbol{\lambda}_0 \geq \mathbf{0}$, $\nabla L(\mathbf{x}_0, \boldsymbol{\lambda}_0) = \mathbf{0}$, and the complementary slackness conditions hold. This result demonstrates that the regularity condition in the hypothesis of Theorem 8.1.1 is important for guaranteeing the uniqueness of the multiplier vectors.

4. Suppose in (1a) that the right-hand side of the second constraint increases from 1 to 1.1. By approximately how much will the optimal objective value for the new NLP differ from that of the original problem? Answer this question without actually solving the new NLP.

5. Consider an LP written in matrix inequality form as

$$\begin{aligned} \text{maximize } & z = \mathbf{c}^t \cdot \mathbf{x} \\ \text{subject to} & \\ & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{8.17}$$

Here $\mathbf{x}$ and $\mathbf{c}$ belong to $\mathbb{R}^n$, A is an m-by-n matrix, and $\mathbf{b}$ belongs to $\mathbb{R}^m$.

(a) Verify that the constraints and sign conditions can be combined into the single matrix inequality

$$\begin{bmatrix} A \\ -I_n \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_{n\times 1} \end{bmatrix} \leq \mathbf{0}_{(n+m)\times 1}. \tag{8.18}$$

(b) Viewing the LP as an NLP, construct its Lagrangian function, $L(\mathbf{x}, \boldsymbol{\lambda})$, along with $\nabla L(\mathbf{x}, \boldsymbol{\lambda})$, in terms of the given vector and matrix quantities.

(c) Show that if $\mathbf{x}_0$ is an optimal solution of (8.18) then the first m entries of the Lagrange multiplier vector, $\boldsymbol{\lambda}_0$, form the solution of the corresponding dual LP. (Hint: Combine the Karush-Kuhn-Tucker Theorem with the Complementary Slackness Theorem, Theorem 4.1.4.)

(d) Verify the preceding result by applying it to the *FuelPro* LP, (4.2). In this case, $\boldsymbol{\lambda}_0$ belongs to $\mathbb{R}^5$.

(e) The first three entries of $\boldsymbol{\lambda}_0$ in (d) form the solution of the dual of (8.18). What is the meaning of the remaining two entries?

6. Recall *Pam's Pentathlon Training Program* NLP, as discussed in Exercise 2, from Section 6.1.

$$\begin{aligned} \text{maximize } f(x_1, x_2, x_3) &= .11193(254 - (180 - .5x_1 - x_2 - x_3))^{1.88} \\ &\quad + 56.0211(5.2 + .1x_1)^{1.05} \\ \text{subject to} \\ & 6 \leq x_1 + x_2 + x_3 \leq 10 \\ & 2 \leq x_1 \leq 4 \\ & 3 \leq x_2 \leq 4 \\ & .6x_1 - .4x_3 \leq 0 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

The decision variables, x_1, x_2, and x_3, denote the total number of hours per week Pam devotes to weight lifting, distance running, and speed workouts, and the constraints give requirements as to how Pam allots her training time. The objective function represents the portion of Pam's total pentathlon score stemming from her performances in the 800 meter run and shot put. It is based upon International Amateur Athletic Federation scoring formulas, together with the fact that Pam currently completes the 800 meter run in 3 minutes and throws the shot put 6.7 meters. The NLP has a total of ten constraints, including sign restrictions.

(a) The solution to this NLP is given by $x_1 = 2.5$, $x_2 = 3$, and $x_3 = 4.5$ hours. Use this information to determine the values of the ten corresponding Lagrange multipliers. (Hint: First determine which constraints are nonbinding at the solution. The result, by complementary slackness, quickly indicates a subset of multipliers equalling zero.)

(b) If Pam has 30 minutes of more available training time each week, by approximately how much will her score increase? Answer this question using your result from (a).

8.2 Convex NLPs

Theorem 8.1.1 is useful insofar as guaranteeing the existence of Lagrange multipliers corresponding to the solution of NLP (8.3) under appropriate hypotheses. It gives necessary conditions that must hold at a minimum, but these conditions alone are not sufficient. The following Waypoint provides an example demonstrating this fact.

◆ ◆

Waypoint 8.2.1. Show that $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ is a regular KKT point, yet not a solution of the NLP

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = x_1^2 - x_2^2 \\ &\text{subject to} \\ &\qquad g_1(\mathbf{x}) = x_1 \leq 0 \\ &\qquad g_2(\mathbf{x}) = x_2 \leq 0. \end{aligned}$$

Then explain why the NLP is unbounded.

◆ ◆

8.2.1 Solving Convex NLPs

The previous Waypoint illustrates that a regular KKT point, $\mathbf{x}_0$, is merely a candidate for a solution. Different means exist for determining whether $\mathbf{x}_0$ is also a solution NLP (8.3). One such tool involves convexity. Recall from Theorem 6.3.1 that if $S \subseteq \mathbb{R}^n$ is convex and $f : S \to \mathbb{R}$ is differentiable at each point in S, then f is convex on S if and only if for every $\mathbf{x}_0$ in S,

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) \text{ for all } \mathbf{x} \in S. \tag{8.19}$$

This result, which arose in the context of unconstrained NLPs, is essential for proving the following theorem.

Theorem 8.2.1. Suppose $S \subseteq \mathbb{R}^n$ is convex and that the function f from the NLP (8.3), as well as each component of $\mathbf{g}$, are continuously differentiable on S. Assume that f is convex on S, as is each component, g_i, of $\mathbf{g}$. Suppose $\mathbf{h}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$ for some p by n matrix A and some vector $\mathbf{b}$ in $\mathbb{R}^p$. Then any regular KKT point, $\mathbf{x}_0$, of the NLP is a solution.

Remark: An NLP of the form described above is called a *convex* NLP.

Proof. By Theorem 8.1.1, there exist multiplier vectors, $\boldsymbol{\lambda}_0$ in $\mathbb{R}^m$ and $\boldsymbol{\mu}$ in $\mathbb{R}^p$ satisfying conditions (8.9), (8.10), and (8.11). Reordering the inequality constraints if necessary, we may assume that the first k constraints, where $1 \leq k \leq m$, are binding at $\mathbf{x}_0$. Hence $g_i(\mathbf{x}_0) = 0$ for $1 \leq i \leq k$, and, by complementary slackness, $\lambda_{0,i} = 0$ for $k+1 \leq i \leq m$.

Each function g_i is differentiable and convex on S; therefore, Theorem 6.3.1 yields, for $1 \leq i \leq k$ and for all $\mathbf{x} \in S$,

$$\begin{aligned} \nabla g_i(\mathbf{x}_0)^t(\mathbf{x} - \mathbf{x}_0) &\leq g_i(\mathbf{x}) - g_i(\mathbf{x}_0) \\ &= g_i(\mathbf{x}) \\ &\leq 0. \end{aligned} \tag{8.20}$$

Because $\lambda_{0,i} \geq 0$ for all $1 \leq i \leq k$ and $\lambda_{0,i} = 0$ for $k+1 \leq i \leq m$,

$$\begin{aligned} \boldsymbol{\lambda}_0^t \mathbf{J_g}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) &= \left(\sum_{i=1}^{m} \lambda_{0,i} \nabla g_i(\mathbf{x}_0)^t \right) (\mathbf{x} - \mathbf{x}_0) \\ &= \sum_{i=1}^{m} \lambda_{0,i} \left(\nabla g_i(\mathbf{x}_0)^t (\mathbf{x} - \mathbf{x}_0) \right) \\ &\leq 0. \end{aligned} \tag{8.21}$$

For equality constraints, we note that $\mathbf{J_h}(\mathbf{x}) = A$. Hence, for feasible $\mathbf{x}$,

$$\begin{aligned} \boldsymbol{\mu}_0^t \mathbf{J_h}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) &= \boldsymbol{\mu}_0^t (A\mathbf{x} - A\mathbf{x}_0) \\ &= \boldsymbol{\mu}_0^t \left((A\mathbf{x} - \mathbf{b}) - (A\mathbf{x}_0 - \mathbf{b}) \right) \\ &= 0. \end{aligned} \tag{8.22}$$

We now combine the results of (8.21) and (8.22), together with (8.10), which states

$$\nabla f(\mathbf{x}_0) + \mathbf{J_g}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 + \mathbf{J_h}(\mathbf{x}_0)^t \boldsymbol{\mu}_0 = \mathbf{0}.$$

Fix $\mathbf{x}$ in S. Then, by the convexity of f, along with inequalities (8.21) and (8.22), we obtain the following:

$$\begin{aligned} f(\mathbf{x}) - f(\mathbf{x}_0) &\geq \nabla f(\mathbf{x}_0)^t (\mathbf{x} - \mathbf{x}_0) \\ &= -\left(\mathbf{J_g}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 + \mathbf{J_h}(\mathbf{x}_0)^t \boldsymbol{\mu}_0 \right)^t (\mathbf{x} - \mathbf{x}_0) \\ &= -\left(\boldsymbol{\lambda}_0^t \mathbf{J_g}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \boldsymbol{\mu}_0^t \mathbf{J_h}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \right) \\ &= -\boldsymbol{\lambda}_0^t \mathbf{J_g}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \\ &\geq 0. \end{aligned}$$

Thus $f(\mathbf{x}_0) \leq f(\mathbf{x})$ for all $\mathbf{x} \in S$, so $\mathbf{x}_0$ is a solution of NLP (8.3). □

Theorem 8.2.1 provides the tools for solving a wide range of problems, including the *ConPro Manufacturing Company* NLP:

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = -1400x_1^{\frac{1}{2}}x_2^{\frac{1}{3}} + 350x_1 + 200x_2 \\ \text{subject to} & \\ & g_1(\mathbf{x}) = 350x_1 + 200x_2 - 40000 \leq 0 \\ & g_2(\mathbf{x}) = x_1 - 3x_2 \leq 0 \\ & g_3(\mathbf{x}) = -x_1 \leq 0 \\ & g_4(\mathbf{x}) = -x_2 \leq 0. \end{aligned} \tag{8.23}$$

In this case, $S = \mathbb{R}^2_+ = \{(x_1, x_2) \mid x_1, x_2 > 0\}$. Recall in Section 6.4 that we established f is convex on S by using Theorem 6.4.3. Each constraint function, $g_i(\mathbf{x})$, from (8.23) is an *affine transformation* and, hence, is convex on S. (See Section 6.3.1). Thus (8.23) is a convex NLP, so finding its solution reduces to finding its regular KKT points.

At $\mathbf{x}_0$, the gradient of the Lagrangian must vanish. Hence,

$$\begin{aligned} \nabla L(\mathbf{x}_0) &= \nabla f(\mathbf{x}_0) + \sum_{i=1}^{4} \lambda_i \nabla g_i(\mathbf{x}_0) \\ &= 0. \end{aligned} \tag{8.24}$$

The complementary slackness conditions require that

$$\begin{aligned} \lambda_1 g_1(\mathbf{x}_0) &= 0 \\ \lambda_2 g_2(\mathbf{x}_0) &= 0 \\ \lambda_3 g_3(\mathbf{x}_0) &= 0 \\ \lambda_4 g_4(\mathbf{x}_0) &= 0. \end{aligned} \tag{8.25}$$

That $\nabla L(\mathbf{x}) = \mathbf{0}$ provides two equations; complementary slackness provides the remaining four. Hence we must solve a system of six equations in six unknowns, $x_1, x_2, \lambda_1, \lambda_2, \lambda_3, \lambda_4$.

This task is best accomplished using Maple. Here is a sample worksheet, **Computing KKT Points.mw**, illustrating how to carry out this process:

```
> with(VectorCalculus):with(LinearAlgebra):
> f:=(x1,x2)-> -1400*x1^(1/2)*x2^(1/3)+350*x1+200*x2:
  # Enter objective function.
> g1:=(x1,x2)->350*x1+200*x2-40000:
```

```
> g2:=(x1,x2)->x1-3*x2:
> g3:=(x1,x2)->-x1:
> g4:=(x1,x2)->-x2:
> g:=(x1,x2)-><g1(x1,x2),g2(x1,x2),g3(x1,x2),g4(x1,x2)>:g(x1,x2);
  # Enter vector-valued constraint function.
```

$$(350x_1 + 200x_2 - 40000)\mathbf{e}_{x_1} + (x_1 - 3x_2)\mathbf{e}_{x_2} - x_1\mathbf{e}_{x_3} - x_2\mathbf{e}_{x_4}$$

```
> lambda:=<lambda1,lambda2,lambda3,lambda4>:
  # Create vector of multipliers.
> L:=unapply(f(x1,x2)+Transpose(lambda).g(x1,x2),
  [x1,x2,lambda1,lambda2,lambda3,lambda4]):
  # Create Lagrangian Function.
> LG:=Gradient(L(x1,x2,lambda1,lambda2,lambda3,lambda4),
  [x1,x2,lambda1,lambda2,lambda3,lambda4]):
  # Create Lagrangian Gradient.
> CS:=seq(g(x1,x2)[i]*lambda[i]=0,i=1..4):
  # Create complementary slackness equations.
> solve({CS, LG[1]=0,LG[2]=0 }, {x1,x2,lambda1,lambda2,lambda3,lambda4});
  # Solve system of equations to determine KKT point and multipliers.
```

We have omitted the output produced by the last command, due to the fact Maple returns several solutions. However, only one of these satisfies the feasibility conditions along with the sign restriction on the multipliers. The resulting values are $x_1 = \dfrac{480}{7} \approx 68.57$, $x_2 = 80$, $\lambda_1 = \sqrt{\dfrac{7}{120}}\sqrt[3]{80} - 1 \approx .04$, and $\lambda_2 = \lambda_3 = \lambda_4 = 0$. Thus the sole KKT point is $\mathbf{x}_0 = \begin{bmatrix} \frac{480}{7} \\ 80 \end{bmatrix} \approx \begin{bmatrix} 68.57 \\ 80 \end{bmatrix}$. Only the first constraint, $g_1(\mathbf{x}) = 350x_1 + 200x_2 - 40,000 \leq 0$, is binding at $\mathbf{x}_0$. Thus, the regularity condition is trivially satisfied, so that $\mathbf{x}_0$ is a regular KKT point and, therefore, is the solution of the *ConPro Manufacturing Company*NLP (8.23).

In this section we have developed a tool for solving a wide range of NLPs. In the next section, we develop means of solving certain non-convex NLPs.

Exercises Section 8.2

1. For each of the following NLPs, verify that the problem is convex and then determine its solution.

(a)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1^2 - 6x_1 + x_2^2 - 4x_2 \\ \text{subject to} & \\ & x_1 + x_2 \leq 4 \\ & x_1 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

(b)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = (x_1 - 6)^2 + (x_2 - 4)^2 \\ \text{subject to} & \\ & -x_1 + x_2^2 \leq 0 \\ & x_1 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

(c)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1^2 - \ln(x_2 + 1) \\ \text{subject to} & \\ & 2x_1 + x2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

(d)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = 4x_1^2 - 12x_1 + x_2^2 - 6x_2 \\ \text{subject to} & \\ & -x_1 + x_2^2 \leq 2 \\ & 2x_1 + x_2 = 1 \end{aligned}$$

(e)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = \frac{1}{x_1 x_2} \\ \text{subject to} & \\ & x_1 + 2x_2 = 3 \\ & x_1, x_2 > 0 \end{aligned}$$

2. Suppose that A is an n-by-n matrix, and define $f : \mathbb{R}^n \to \mathbb{R}$ by $f(\mathbf{x}) = \mathbf{x}^t A\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$. Consider the NLP

$$\begin{aligned} \text{minimize } & f(\mathbf{x}) \\ \text{subject to} & \\ & \|\mathbf{x}\| \leq 1 \end{aligned}$$

Show that the optimal solution occurs at the origin, unless A possesses a positive eigenvalue. Show that in this latter case, the solution occurs at the eigenvector, $\mathbf{x}_0$, which is normalized to have length one and which corresponds to the largest eigenvalue, λ. Verify that the corresponding objective value is given by $f(\mathbf{x}_0) = -\lambda$.

3. A company produces a certain product at two different factories and delivers the finished product to its two retail outlets.[1] Assume that the product is of the type that can be produced in non-integer quantities. At factory # 1, the cost of producing s units, in dollars, is given by $P_1(s) = s^2$; at factory # 2 the corresponding cost formula is $P_2(s) = .8s^2$. The cost to ship one product unit from factory # 1 to outlet # 1 is \$6 and from factory # 2 to outlet # 1 is \$9. The corresponding costs for outlet # 2 are \$8 and \$10, respectively. Assume that the demand at each outlet is at least 20 units of product. If the company wishes to minimize the sum of its production and shipping costs, how many units of the product should be shipped from each factory to each outlet?

[1] Based upon Sharp, et al., [44], (1970)

8.3 Saddle Point Criteria

Unfortunately, many NLPs fail to be convex, in which case a KKT point, $\mathbf{x}_0$, need not be an optimal solution. For example, the NLP given by

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = x_1^2 - 6x_1 + x_2^2 - 4x_2 \\ &\text{subject to} \\ &\quad g_1(\mathbf{x}) = x_1^2 + x_2^2 - 9 \leq 0 \\ &\quad g_2(\mathbf{x}) = x_1^2 - x_2^2 \leq 0 \\ &\quad g_3(\mathbf{x}) = x_2 - 2 \leq 0 \\ &\quad g_4(\mathbf{x}) = 2x_1 + x_2 - 8 \leq 0, \end{aligned} \tag{8.26}$$

is a non-convex NLP since g_2 is not a convex function. Straightforward calculations, which we leave as an exercise, show that (8.26) has two KKT points, $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, along with multiplier vector, $\boldsymbol{\lambda}_0 = \begin{bmatrix} 0 \\ 1/2 \\ 2 \\ 0 \end{bmatrix}$, and $\mathbf{x}_0 = \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix}$ together with $\boldsymbol{\lambda}_0 = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 0 \end{bmatrix}$. As we shall discover, only the first of these KKT points forms a solution to (8.26).

8.3.1 The Restricted Lagrangian

Throughout the ensuing discussion, we assume that $\mathbf{x}_0$ is a KKT point of (8.1) having corresponding multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$. We may assume, by relabeling if necessary, that the first k components of $\mathbf{g}$, where $1 \leq k \leq m$, are *binding* at $\mathbf{x}_0$. By this, we mean that $g_i(\mathbf{x}_0) = 0$ for $1 \leq i \leq k$ and $g_i(\mathbf{x}_0) < 0$ for $k + 1 \leq i \leq m$. With components of $\mathbf{g}$ labeled in this manner, we define the vector-valued function, $\widetilde{\mathbf{g}} : \mathbb{R}^n \to \mathbb{R}^m$ given by $\widetilde{\mathbf{g}}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_k(\mathbf{x}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$.

Note that $\widetilde{\mathbf{g}}(\mathbf{x}_0) = \mathbf{0}$ and that, due to complementary slackness conditions, $\lambda_{0,i} = 0$ for $k + 1 \leq i \leq m$.

Definition 8.3.1. The *restricted Lagrangian function* corresponding to a KKT point, $\mathbf{x}_0$, of NLP (8.1) is the function

$$\begin{aligned}\widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f(\mathbf{x}) + \boldsymbol{\lambda}^t \widetilde{\mathbf{g}}(\mathbf{x}) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}) \\ &= f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x}),\end{aligned} \tag{8.27}$$

where $\mathbf{x}$ belongs to S, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are vectors in $\mathbb{R}^m$ and $\mathbb{R}^p$, respectively, with $\boldsymbol{\lambda} \geq \mathbf{0}$.

◆ ——— ◆

Waypoint 8.3.1. Calculate the restricted Lagrangian for NLP (8.26) at each of its two KKT points.

◆ ——— ◆

To understand how the restricted Lagrangian compares to the original, note that if $\mathbf{x}$ is feasible, then $\lambda_i g_i(\mathbf{x}) \leq 0$ for $\lambda_i \geq 0$ and $1 \leq i \leq m$, so that

$$\begin{aligned}\widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f(\mathbf{x}) + \boldsymbol{\lambda}^t \widetilde{\mathbf{g}}(\mathbf{x}) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}) \\ &= f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x}) \\ &\geq f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x}) \\ &= L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}).\end{aligned}$$

Furthermore, suppose $\mathbf{x}_0$ is a KKT point of NLP (8.1) with corresponding multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$. Since $\lambda_{0,i} = 0$ for $k+1 \leq i \leq m$, $\mathbf{J}_{\widetilde{\mathbf{g}}}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 = \mathbf{J}_{\mathbf{g}}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0$. Therefore,

$$\begin{aligned}\nabla \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) &= \nabla f(\mathbf{x}_0) + \mathbf{J}_{\widetilde{\mathbf{g}}}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 + \mathbf{J}_{\mathbf{h}}(\mathbf{x}_0)^t \boldsymbol{\mu}_0 \\ &= \nabla f(\mathbf{x}_0) + \mathbf{J}_{\mathbf{g}}(\mathbf{x}_0)^t \boldsymbol{\lambda}_0 + \mathbf{J}_{\mathbf{h}}(\mathbf{x}_0)^t \boldsymbol{\mu}_0 \\ &= \nabla L(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \\ &= \mathbf{0}.\end{aligned} \tag{8.28}$$

Thus $\mathbf{x}_0$ is a KKT point of the $\widetilde{L}$ with multiplier vectors $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$.

The restricted Lagrangian, $\widetilde{L}$ has the special property that if $\mathbf{x}_0$ is a KKT point of NLP (8.1) with corresponding multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$, then the gradient of $\widetilde{L}$ with respect to all three vector variables, $\mathbf{x}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\mu}$, must vanish

at $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$. In other words, $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ is a critical point of $\widetilde{L}$. To see why this is so, first note from (8.28) that

$$\left.\frac{\partial \widetilde{L}}{\partial x_k}\right|_{(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)} = 0 \quad \text{for } 1 \le k \le n. \tag{8.29}$$

Moreover, since $\mathbf{x}_0$ is a feasible solution of NLP (8.1), constraints $g_i(\mathbf{x})$ are binding at $\mathbf{x}_0$ for $i = 1, 2, \ldots, k$, so that

$$\begin{aligned}\left.\frac{\partial \widetilde{L}}{\partial \lambda_i}\right|_{(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)} &= g_i(\mathbf{x}_0) \\ &= 0 \qquad \text{for } 1 \le i \le k.\end{aligned}$$

This result, along with the fact $\widetilde{L}$ is independent of $\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_m$, leads to

$$\left.\frac{\partial \widetilde{L}}{\partial \lambda_i}\right|_{(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)} = 0 \quad \text{for } 1 \le i \le m. \tag{8.30}$$

Finally,

$$\begin{aligned}\left.\frac{\partial \widetilde{L}}{\partial \mu_j}\right|_{(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)} &= h_j(\mathbf{x}_0) \\ &= 0 \qquad \text{for } 1 \le j \le p.\end{aligned} \tag{8.31}$$

Combining (8.29), (8.30), and (8.31), we conclude $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = (\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ is a critical point of $\widetilde{L}$.

8.3.2 Saddle Point Optimality Criteria

That the restricted Lagrangian, $\widetilde{L}$, has a critical point at $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ proves quite useful for establishing $\mathbf{x}_0$ is a solution of NLP (8.1). The specific conditions for which this occurs are phrased in terms of the following definition.

Definition 8.3.2. Suppose the triple $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ is a critical point of the restricted Lagrangian, $\widetilde{L}$, where $\mathbf{x}_0$ is a KKT point of (8.1) with multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$, where $\boldsymbol{\lambda}_0 \ge \mathbf{0}$. Then we say that $\widetilde{L}$ satisfies the *saddle point criteria* at $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ if and only if

$$\widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}, \boldsymbol{\mu}) \le \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \le \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \tag{8.32}$$

for all $\mathbf{x}$ in S, all $\boldsymbol{\lambda}$ in $\mathbb{R}^m$ satisfying $\boldsymbol{\lambda} \ge \mathbf{0}$, and all $\boldsymbol{\mu}$ in $\mathbb{R}^p$.

We leave it as an exercise to show that for a convex NLP, any triple, $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$, where $\mathbf{x}_0$ is a KKT point with corresponding multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$, is a saddle point of $\widetilde{L}$. Thus a KKT point for a convex NLP is a solution of the NLP and also corresponds to a saddle point of the restricted Lagrangian.

The following result is significant in that it applies to the non-convex setting and shows how a KKT point of an NLP, which corresponds to a saddle point of the restricted Lagrangian, yields a solution of the NLP.

Theorem 8.3.1. Suppose $\mathbf{x}_0$ is a KKT point of (8.1) with multiplier vectors, $\boldsymbol{\lambda}_0$ and $\boldsymbol{\mu}_0$. If $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ is a saddle point of $\widetilde{L}$, then $\mathbf{x}_0$ is a solution of the NLP.

Proof. Assume $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ is a saddle point of $\widetilde{L}$ and that $\mathbf{x}$ in S is feasible. Then

$$\begin{aligned} f(\mathbf{x}_0) &= f(\mathbf{x}_0) + \boldsymbol{\lambda}^t \widetilde{\mathbf{g}}(\mathbf{x}_0) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}_0) \qquad (8.33)\\ &= \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &\leq \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \\ &\leq \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}_0^t \widetilde{\mathbf{g}}(\mathbf{x}) + \boldsymbol{\mu}_0^t \mathbf{h}(\mathbf{x}) \\ &\leq f(\mathbf{x}) \qquad (8.34) \end{aligned}$$

Thus $\mathbf{x}_0$ is a solution of NLP (8.1). □

To test whether a KKT point and its multiplier vectors satisfy the saddle point criteria, we first define $\phi(\mathbf{x}) = \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ and establish that $\phi(\mathbf{x}_0) \leq \phi(\mathbf{x})$ for all feasible $\mathbf{x}$ in S. For when this holds, feasibility requirements along with the condition $\boldsymbol{\lambda} \geq \mathbf{0}$, yield

$$\begin{aligned} \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) &\geq \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0) \\ &= f(\mathbf{x}_0) + \boldsymbol{\lambda}_0^t \widetilde{\mathbf{g}}(\mathbf{x}_0) + \boldsymbol{\mu}_0^t \mathbf{h}(\mathbf{x}_0) \\ &\geq f(\mathbf{x}_0) + \boldsymbol{\lambda}^t \widetilde{\mathbf{g}}(\mathbf{x}_0) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}_0) \\ &= \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}, \boldsymbol{\mu}), \end{aligned}$$

which is precisely what it means to say that $(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ is a saddle point of $\widetilde{L}$.

We can apply this technique to the non-convex NLP (8.26). At $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, the

second and third constraints are binding. Since $\boldsymbol{\lambda}_0 = \begin{bmatrix} 0 \\ 1/2 \\ 2 \\ 0 \end{bmatrix}$, we obtain

$$\begin{aligned} \phi(\mathbf{x}) &= \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0) \\ &= f(\mathbf{x}) + \frac{1}{2} g_2(\mathbf{x}) + 2g_3(\mathbf{x}) \\ &= \frac{3}{2}x_1^2 - 6x_1 + \frac{1}{2}x_2^2 - 2x_2 - 4. \end{aligned}$$

An elementary exercise establishes that $\nabla\phi$ vanishes precisely at $\mathbf{x}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ and that the Hessian of ϕ is constant and positive definite. Thus, ϕ has a global minimum at $\mathbf{x}_0$ so that $\mathbf{x}_0$ is a solution of NLP (8.26).

At the second KKT point, $\mathbf{x}_0 = \begin{bmatrix} 1/2 \\ -1/2 \end{bmatrix}$, only the second constraint is binding. In this case, $\phi(\mathbf{x}_0) = x_1^2 - x_2^2$ has a saddle point at $\mathbf{x}_0$. Thus, we may not conclude $\mathbf{x}_0$ is also a solution of NLP (8.26). In fact, $f\left(\frac{1}{2}, -\frac{1}{2}\right) = -.5$, whereas $f(2,2) = -12$.

Exercises Section 8.3

1. For each of the following NLPs, explain why the NLP is not convex. Then determine the KKT points and corresponding multiplier vectors. Finally, verify that the restricted Lagrangian satisfies the saddle point criteria. (Note: Determining the KKT points and corresponding multiplier vectors can be accomplished using Maple as outlined at the end of Section 8.2.1.)

   (a)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1^2 - x_2^2 \\ \text{subject to} \\ & x_1 - x_2 \leq 0 \\ & x_1^2 + x_2^2 \leq 4 \\ & x_2 \geq 0 \end{aligned}$$

(b)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_1 x_2 \\ \text{subject to} & \\ & x_1^2 + x_2^2 \leq 1 \\ & 2x_1 + 3x_2 \leq 3 \\ & x_1 \geq 0 \end{aligned}$$

(c)

$$\begin{aligned} \text{minimize } & f(x_1, x_2) = x_2^2 - 6x1 + x_2^2 - 4x_2 \\ \text{subject to} & \\ & x_1^2 + x_2^2 \leq 9 \\ & x_1^2 - x_2^2 \leq 0 \\ & 2x_1 + x_2 \leq 8 \\ & x_2 \leq 2 \end{aligned}$$

2. Verify that the restricted Lagrangian for the *ConPro Manufacturing Company*NLP satisfies the saddle point criteria at its optimal solution.

3. Consider the convex NLP

$$\begin{aligned} \text{minimize } & f(\mathbf{x}) \\ \text{subject to} & \\ & g_1(\mathbf{x}) \leq 0 \\ & g_2(\mathbf{x}) \leq 0 \\ & \vdots \\ & g_m(\mathbf{x}) \leq 0, \end{aligned}$$

where $f, g_1, g_2, \ldots, g_m$ are each differentiable, convex functions defined on some convex set $S \subseteq \mathbb{R}^n$. In this problem, we show that if $\mathbf{x}_0$ is a KKT point with corresponding multiplier vector, $\boldsymbol{\lambda}_0$, then $(\mathbf{x}_0, \boldsymbol{\lambda}_0)$ is a saddle point of the restricted Lagrangian. In other words, we establish the two inequalities that comprise (8.32).

   (a) Suppose that $\boldsymbol{\lambda}$ is a vector in $\mathbb{R}^m$ having nonnegative components λ_i, where $1 \leq i \leq m$. To show that $\widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}) \leq \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0)$, first express the difference $\widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}, \boldsymbol{\mu}) - \widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0)$ in terms of $f, g_1, g_2, \ldots, g_m$ and the components of $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}_0$.

(b) To establish that $\widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0) \leq \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0)$, start by expressing the difference $\widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0) - \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0)$ in terms of $f, g_1, g_2, \ldots, g_m$ and the Lagrange multipliers, $\lambda_{0,1}, \lambda_{0,2}, \ldots \lambda_{0,m}$.

(c) Now apply Theorem 6.3.1 to each of the functions $f, g_1, g_2, \ldots, g_m$ from (c) in order to show that

$$\widetilde{L}(\mathbf{x}_0, \boldsymbol{\lambda}_0) - \widetilde{L}(\mathbf{x}, \boldsymbol{\lambda}_0) \leq -\left(\nabla f(\mathbf{x}_0)^t + \mathbf{J}_{\mathbf{g}}(\mathbf{x}_0)^t \lambda_0\right)(\mathbf{x} - \mathbf{x}_0).$$

(d) Finally, apply Equation (8.10) to the right-hand side of the previous inequality.

8.4 Quadratic Programming

Many real-world nonlinear programming problems are quadratic in nature. In this section we develop develop a framework for solving such problems.

8.4.1 Problems with Equality-type Constraints Only

A quadratic programming problem is written in matrix form as follows:

$$\begin{aligned} \text{minimize } & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x} + \mathbf{p}^t\mathbf{x} \\ \text{subject to} & \\ & A\mathbf{x} = \mathbf{b} \\ & C\mathbf{x} \leq \mathbf{d}, \end{aligned} \tag{8.35}$$

where $\mathbf{x}$ belongs to $\mathbb{R}^n$, Q is a symmetric n-by-n matrix, A is an m-by-n matrix, C is a p-by-n matrix, and $\mathbf{b}$ and $\mathbf{d}$ belong to $\mathbb{R}^m$ and $\mathbb{R}^p$, respectively.

As a first step towards solving (8.35), we consider the special case when it has only equality-type constraints. In other words, we seek to solve

$$\begin{aligned} \text{minimize } & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x} + \mathbf{p}^t\mathbf{x} \\ \text{subject to} & \\ & A\mathbf{x} = \mathbf{b}. \end{aligned} \tag{8.36}$$

We will assume that the system of equations $A\mathbf{x} = \mathbf{b}$ is consistent and that no single equation comprising the system is a linear combination of the others. Therefore, the rows of A must form a linearly independent set of vectors, so that $m \leq n$ and any KKT point is regular.

The Lagrangian corresponding to (8.36) is given by

$$L(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\mu}^t (A\mathbf{x} - \mathbf{b}) \text{ where } \boldsymbol{\mu} \in \mathbb{R}^p. \tag{8.37}$$

By Theorem 8.1.1, if $\mathbf{x}_0$ is a solution of (8.36), then it must be feasible, i.e., $A\mathbf{x}_0 = \mathbf{b}$, and satisfy

$$\begin{aligned} \mathbf{0} &= \nabla L(\mathbf{x}_0) \\ &= \nabla f(\mathbf{x}) + A^t\boldsymbol{\mu}_0 \\ &= Q\mathbf{x}_0 + \mathbf{p} + A^t\boldsymbol{\mu}_0 \end{aligned} \tag{8.38}$$

for some $\boldsymbol{\mu}_0$ in $\mathbb{R}^p$.

These conditions may be conveniently combined into the form of a partitioned matrix equation,

$$\begin{bmatrix} 0_{m\times m} & A \\ A^t & Q \end{bmatrix}\begin{bmatrix} \boldsymbol{\mu}_0 \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{p} \end{bmatrix}. \tag{8.39}$$

In the ideal situation, the matrix Q is positive definite. When this occurs (8.36) is a convex NLP so that any solution of (8.39) yields a regular KKT point, $\mathbf{x}_0$, which is a solution of (8.36) by Theorem (8.2.1).

When Q is positive definite, the coefficient matrix in (8.39) is invertible. In particular, if we denote $S = -AQ^{-1}A^t$, the inverse is given by

$$\begin{bmatrix} 0_{m\times m} & A \\ A^t & Q \end{bmatrix}^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}AQ^{-1} \\ -Q^{-1}A^tS^{-1} & Q^{-1}(Q + A^tS^{-1}A)Q^{-1} \end{bmatrix}. \tag{8.40}$$

The matrix S is called the *Schur Complement* of Q in the coefficient matrix in (8.39). That S itself is invertible can be proven using the Spectral Theorem (Theorem B.7.1).

For example, suppose $Q = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 4 & 1 \\ 0 & 1 & 5 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 2 \\ -3 \\ 0 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 0 & 0 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. The eigenvalues of Q are positive, so Q is positive definite. The solution of (8.39) is given by

$$\begin{bmatrix} \boldsymbol{\mu}_0 \\ \mathbf{x}_0 \end{bmatrix} \approx \begin{bmatrix} .3929 \\ -5.5357 \\ 2 \\ -.4286 \\ -.0714 \end{bmatrix}$$

with corresponding objective value $f(\mathbf{x}_0) \approx 7.9821$.

When Q is not positive definite, the situation becomes somewhat more complicated. In this case, equation (8.39) still remains valid. However, the coefficient matrix need not be invertible and, even if it is, (8.36) is no longer a convex NLP, so a solution of (8.39) need not yield a solution of the original problem.

Fortunately, a tool exists for circumventing this problem, one that applies to many situations. To understand the rationale behind it, we first consider a simple example. Suppose, in (8.36), that $Q = \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$, $\mathbf{p} = \mathbf{0}$, $A = \begin{bmatrix} 1 & -1 \end{bmatrix}$, and $\mathbf{b} = 1$. (For the sake of consistent notation throughout the discussion, we

express **b** using vector notation even though it is a scalar for this particular problem.) The feasible region for this NLP consists of the line $x_1 - x_2 = 1$ in $\mathbb{R}^2$. However, the eigenvalues of Q are mixed-sign, so the NLP is not convex.

Figure (8.2) depicts the graph of the objective function, f, a saddle-shaped surface, along with the image under f of the line $x_1 - x_2 = 1$, which constitutes a parabola embedded in the surface.

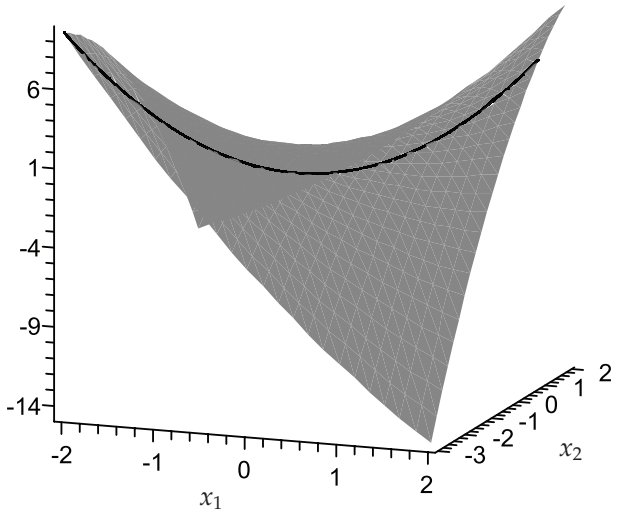


FIGURE 8.2: Quadratic form, f, together with the image under f of the line $x_1 - x_2 = 1$.

Observe that while the graph of f is saddle-shaped, the restriction of f to the line $x_1 - x_2 = 1$ is an upward-opening parabola. In fact, if we substitute $x_2 = x_1 - 1$ into $f(\mathbf{x}) = \mathbf{x}^t Q\mathbf{x}$, we obtain

$$\begin{aligned} f(\mathbf{x}) &= f(x_1, x_1 - 1) \\ &= \begin{bmatrix} x_1 \\ x_1 - 1 \end{bmatrix}^t Q \begin{bmatrix} x_1 \\ x_1 - 1 \end{bmatrix} \\ &= \frac{1}{2}x_1^2 + 2x_1x_2 - \frac{1}{2}x_2^2 \\ &= 2x_1^2 - x1 - \frac{1}{2}, \end{aligned}$$

which is a convex function of x_1 and has a minimum at $x_1 = \frac{1}{4}$. This value corresponds to $x_2 = -\frac{3}{4}$ so that $\mathbf{x}_0 = \begin{bmatrix} 1/4 \\ -3/4 \end{bmatrix}$ is the solution of (8.36). Observe this result can also be obtained by solving (8.39), in which case

$$\begin{bmatrix} \mu_0 \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1/4 \\ -3/4 \end{bmatrix}.$$

Although the calculations are somewhat tedious, we can replicate the previous steps to show that if $Q = \begin{bmatrix} q_1 & q \\ q & q_2 \end{bmatrix}$, $\mathbf{p} = \mathbf{0}$, $A = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$, and $\mathbf{b} \in \mathbb{R}$, then the restriction of $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x}$ to the set of vectors $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ satisfying $A\mathbf{x} = \mathbf{b}$ is a quadratic expression in x_1, whose leading coefficient is given by $\dfrac{q_1 a_2^2 - 2q a_2 a_1 + q_2 a_1^2}{2a_2^2}$. Thus, as a function of x_1 alone, this restriction of f is convex provided $q_1^2 a_2^2 - 2q a_2 a_1 + q_2 a_1^2 > 0$. This was the case in our preceding example, where $q_1 = 1$, $q_2 = -1$, $q = 2$, $a_1 = 1$, and $a_2 = -1$.

The condition $\dfrac{q_1^2 a_2^2 - 2q a_2 a_1 + q_2 a_1^2}{2a_2^2} > 0$ can be restated in terms of a determinant identity. Namely, it holds provided

$$\begin{aligned} -\det\left(\begin{bmatrix} 0 & A \\ A^t & Q \end{bmatrix}\right) &= -\det\left(\begin{bmatrix} 0 & a_1 & a_2 \\ a_1 & q_1 & q \\ a_2 & q & q_2 \end{bmatrix}\right) \\ &= q_1^2 a_2^2 - 2q a_2 a_1 + q_2 a_1^2 \\ &> 0. \end{aligned}$$

The matrix $B = \begin{bmatrix} 0 & A \\ A^t & Q \end{bmatrix}$, which is identical to the coefficient matrix in (8.39), is known as a *bordered Hessian*. It consists of the Hessian of f, Q in this case, bordered above by A and to the left by A^t. In the absence of constraints, B is simply Q. Thus, the bordered Hessian can be viewed as a generalization of the normal Hessian to the constrained setting, one that incorporates useful information regarding the constraints themselves. It is an extremely useful tool for solving the quadratic programming problem (8.36). However, before stating a general result that spells out how this is so, we introduce some new terminology.

Definition 8.4.1. Suppose M is an n-by-n matrix. Then the *leading principal*

minor of order k, where $0 \leq k \leq n$, is the determinant of the matrix formed by deleting the last $n - k$ rows and the last $n - k$ columns of M.

Note that when $k = 0$, the leading principal minor is just M itself. With this terminology, we now state a classic result, known as the *bordered Hessian test*.

Theorem 8.4.1. Assume the quadratic programming problem (8.36) has a KKT point at $\mathbf{x}_0$ with corresponding multiplier vector, $\boldsymbol{\mu}_0$. Let $B = \begin{bmatrix} 0 & A \\ A^t & Q \end{bmatrix}$ denote the corresponding $n + m$ by $n + m$ bordered Hessian formed using Q and A. Then $\mathbf{x}_0$ is a solution of (8.36) if the determinant of B and the last $n - m$ leading principal minors of B all have the same sign as $(-1)^m$.

The proof of this result can be found in a variety of sources [29]. Instead of proving it here, we examine its significance in the context of investigating a particular example.

For example, suppose in (8.36) that $Q = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 2 & 1 \\ 0 & 1 & -1 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$, $A = \begin{bmatrix} 1 & -1 & 2 \end{bmatrix}$, and $\mathbf{b} = 1$.

In this situation, Q is indefinite and the solution to (8.36) is given by

$$\begin{bmatrix} \boldsymbol{\mu}_0 \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} 5/6 \\ -31/24 \\ -25/24 \\ 5/8 \end{bmatrix} \approx \begin{bmatrix} .8333 \\ -1.2917 \\ -1.0417 \\ .625 \end{bmatrix}.$$

The bordered Hessian, B, is the 4-by-4 matrix,

$$B = \begin{bmatrix} 0 & 1 & -1 & 2 \\ 1 & 3 & -1 & 0 \\ -1 & -1 & 2 & 1 \\ 2 & 0 & 1 & -1 \end{bmatrix}.$$

Since $n = 3$ and $m = 1$, we must compute the determinant of B along with its last two leading principal minors. The determinant of B is -24. The leading principal minor corresponding to deletion of the last row and column of B equals -3, and that corresponding to deletion of the last two rows and columns

equals -1. Since all three determinants have the same sign as $(-1)^m = -1$, we conclude for this example that the solution of (8.36) is

$$\mathbf{x}_0 = \begin{bmatrix} -31/24 \\ -25/24 \\ 5/8 \end{bmatrix} \approx \begin{bmatrix} -1.2917 \\ -1.0417 \\ .625 \end{bmatrix},$$

with corresponding multiplier $\mu_0 = \dfrac{5}{6} \approx .8333$.

8.4.2 Inequality Constraints

We now consider the case when inequality constraints are present in (8.35). Assume that $\mathbf{x}_0$ is a KKT point for this problem. Each row of the p-by-n matrix C corresponds to a constraint. Assume k of these constraints are binding. Let $\tilde{C}$ be the submatrix of C consisting of these k rows, and let $\tilde{\mathbf{d}}$ denote the vector formed using the corresponding k entries of $\mathbf{d}$. Note that

$$\begin{bmatrix} A \\ \tilde{C} \end{bmatrix} \mathbf{x}_0 = \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{d}} \end{bmatrix}.$$

We assume that the rows of $\begin{bmatrix} A \\ \tilde{C} \end{bmatrix}$ form a linear independent set of vectors, implying, as in the previous section, that $\mathbf{x}_0$ is regular and $m + k \leq n$.

An extension of Theorem 8.4.1, whose proof we also omit, uses the matrix Q "bordered" by $\begin{bmatrix} A \\ \tilde{C} \end{bmatrix}$ and its transpose. Specifically, we form the $(m + k + n)$ by $(m + k + n)$ bordered Hessian

$$B = \begin{bmatrix} 0_{m\times m} & 0_{m\times k} & A \\ 0_{k\times m} & 0_{k\times k} & \tilde{C} \\ A^t & \tilde{C}^t & Q \end{bmatrix}.$$

If the last $n - (m + k)$ leading principal minors of B have the same sign as $(-1)^{m+k}$, then $\mathbf{x}_0$ is a solution of (8.35).

To illustrate this solution process, we solve

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x} + \mathbf{p}^t\mathbf{x} \\ &\text{subject to} \\ &\qquad A\mathbf{x} = \mathbf{b} \\ &\qquad C\mathbf{x} \leq \mathbf{d}, \end{aligned} \tag{8.41}$$

where $Q = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 4 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 2 \\ -3 \\ 0 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$, $\mathbf{b} = 2$, $C = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 1 & 0 \end{bmatrix}$, and $\mathbf{d} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$.

Elementary calculations using methods discussed in Section 8.2.1, establish (8.41) has a KKT point given by $\mathbf{x}_0 = \begin{bmatrix} 5/4 \\ -3/8 \\ 3/4 \end{bmatrix}$, with corresponding multipliers $\boldsymbol{\lambda}_0 = \begin{bmatrix} 5/8 \\ 0 \end{bmatrix}$ and $\mu_0 = -\dfrac{25}{8}$. Using $\mathbf{x}_0$, we see that only the first of the two constraints of the matrix inequality $C\mathbf{x} \leq \mathbf{d}$ is binding, so we set $\tilde{C} = \begin{bmatrix} 1 & 2 & 2 \end{bmatrix}$. The two rows forming $\begin{bmatrix} A \\ \tilde{C} \end{bmatrix}$ are linearly independent, so $\mathbf{x}_0$ is regular. The bordered Hessian is given by

$$B = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 & 0 \\ 0 & 2 & 2 & 4 & 1 \\ 1 & 2 & 0 & 1 & 3 \end{bmatrix} \tag{8.42}$$

Since $n = 3$, $m = k = 1$, $n - (m + k) = 1$ so that we must check that the determinant of B and the leading principal minor corresponding to deletion of the last row and column of B have the same sign as $(-1)^{m+k} = 1$. The values of the two determinants are given by 24 and 4, respectively, so we conclude that $\mathbf{x}_0 = \begin{bmatrix} 5/4 \\ -3/8 \\ 3/4 \end{bmatrix}$ is the solution of (8.41).

8.4.3 Maple's QPSolve Command

Maple's `QPSolve` command, located within the `Optimization` package, provides a convenient means for solving quadratic programming problems. The command accepts arguments in the same format as does `NLPSolve`. However, an error message results if the function to be minimized is not quadratic or if any constraint is neither a linear equation nor a linear inequality. The most convenient way to use this command is to enter constraints in matrix form, in a manner similar to that done for the `LPSolve` command from Section 1.1.3. To solve the quadratic programming problem (8.35) having both equality and inequality constraints, we enter the matrices Q, A, and C along with the vectors $\mathbf{p}$, $\mathbf{b}$, and $\mathbf{d}$ and solve the resulting problem with the command `QPSolve([p,Q],[C,d,A,b])`. Note that the matrix and vector constraints within the brackets are entered in the order of inequality constraints first. To solve (8.35) where only inequality-type constraints are present, we

enter `QPSolve([p,Q],[C,d])` and for the corresponding case of equality-type constraints, we use `QPSolve([p,Q],[NoUserValue,NoUserValue,A,b])`. As an example, consider problem (8.41), where $Q = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 4 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 2 \\ -3 \\ 0 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$, $b = 2$, $C = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 1 & 0 \end{bmatrix}$, and $\mathbf{d} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$.

A Maple worksheet for solving this problem is as follows:

```
> restart:with(LinearAlgebra):with(Optimization):
> Q:=Matrix(3,3,[1,2,0,2,4,1,0,1,3]):
> p:=<2,-3,0>:
> A:=Matrix(1,3,[1,0,1]):
> b:=<2>:
> C:=Matrix(2,3,[1,2,2,4,1,0]):
> d:=<2,5>:
> QPSolve([p,Q],[C,d,A,b]);
```

$$\left[4.3125, \begin{bmatrix} 1.25 \\ -.375 \\ .75 \end{bmatrix}\right]$$

Note that `p`, `b`, and `d` must be entered as vectors using either the <, > notation or the `Vector` command. They may not be entered as matrices or as scalars.

One drawback of the `QPSolve` command is its inability to return the corresponding Lagrange multipliers. To obtain these values, we must use tools introduced in the context of solving the *ConPro* problem at the end of Section 8.2.1 and formulate the Lagrangian function. Here is syntax that demonstrates an efficient means for constructing the Lagrangian corresponding to a quadratic programming problem, such as (8.41):

```
> with(VectorCalculus):with(LinearAlgebra):
> Q:=Matrix(3,3,[1,2,0,2,4,1,0,1,3]):
> p:=Matrix(3,1,[2,-3,0]):
> x:=Matrix(3,1,[x1,x2,x3]):
> f:=unapply(1/2*(Transpose(x).Q.x)[1,1]+(Transpose(p).x)[1,1],[x1,x2,x3]):
> A:=Matrix(1,3,[1,0,1]):
> b:=Matrix(1,1,[2]):
> h:=unapply(convert(A.x-b,Vector),[x1,x2,x3]):
> C:=Matrix(2,3,[1,2,2,4,1,0]):
> d:=Matrix(2,1,[2,5]):
> g:=unapply(convert(C.x-d,Vector),[x1,x2,x3]):
```

```
> lambda:=<lambda1,lambda2>:
> mu:=<mu1>:
> L:=unapply(f(x1,x2,x3)+(Transpose(lambda).g(x1,x2,x3))
  +(Transpose(mu).h(x1,x2,x3),[x1,x2,x3,lambda1,lambda2,mu1]):
```

Observe the index, `[1,1]`, added to `(Transpose(x).Q.x)` in the fifth line of the worksheet. It is required to convert `(Transpose(x).Q.x)`, which Maple views as a 1-by-1 matrix, to a scalar-valued function. In essence, the added index, "removes the brackets" so to speak, from the matrix `(Transpose(x).Q.x)`. For identical reasons, we add brackets, `[1,1]`, immediately after `(Transpose(p).x)`.

Once the Lagrangian, `L`, has been created, KKT points and corresponding multipliers can be computed as was done in the *ConPro* worksheet at the end of Section 8.2.1.

8.4.4 The Bimatrix Game

In Sections 4.1.5 and 6.4.5 we investigated the topic of zero-sum matrix games from the perspectives of duality and saddle points of functions, respectively. A *bimatrix* game is a generalization of the zero-sum game, one in which each player has his or her own payoff matrix.

In this example we assume that Ed's payoff matrix is given by

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \tag{8.43}$$

and Steve's by

$$B = \begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix}. \tag{8.44}$$

As in the zero-sum game, Ed picks a column, say column j. Simultaneously, Steve picks a row, call it row i. Then Steve receives an amount equal to the entry in row i column j of B, and Ed receives an amount equal to the corresponding entry in A. For example, if Steve picks row 1 and Ed chooses column 2, then Ed receives \$2 and Steve receives \$3. (Whether each player receives the money from the other or from some third party is inconsequential.)

A pure strategy Nash equilbrium for the bimatrix game is defined in a manner analogous to that done for the zero-sum case. To determine such equilibria, we consider the four possible ordered pairs formed by the two column choices for Ed and the two row choices for Steve. For example, the combination consisting of Ed choosing column one and Steve choosing row one is not a pure strategy equilibrium. If Ed recognizes that Steve always chooses row one, then Ed can increase his earnings by choosing column two provided

Steve continues to follow his own strategy. Similar reasoning applied to other cases establishes that no pure strategy Nash equilibrium exists.

However, as proven by John Nash in 1951, any bimatrix game is guaranteed to possess at least one mixed strategy equilbrium [34]. We now determine such equilibria for Ed and Steve's contest.

As was the case in Section 4.1, matrix-vector products provide a convenient means for expressing each game move. To Ed's choice of column 2 we associate the vector $\begin{bmatrix}0\\1\end{bmatrix}$ and to Steve's choice of row 1 we associate $\begin{bmatrix}1\\0\end{bmatrix}$. Then Ed and Steve have respective earnings given by

$$\begin{bmatrix}1\\0\end{bmatrix}^t A \begin{bmatrix}0\\1\end{bmatrix} = 2 \text{ and } \begin{bmatrix}1\\0\end{bmatrix}^t B \begin{bmatrix}0\\1\end{bmatrix} = 3.$$

If Steve's equilibrium mixed strategy is represented by the vector $\mathbf{y}_0$ in $\mathbb{R}^2$, whose entries sum to 1, then Ed seeks to determine the value $\mathbf{x}$, call it $\mathbf{x}_0$, whose entries sum to 1 and which maximizes $\mathbf{y}_0^t A\mathbf{x}$. Likewise, Steve wants $\mathbf{y}_0$ to be the value of $\mathbf{y}$ that maximizes $\mathbf{y}^t B\mathbf{x}_0$. If $\mathbf{z}_0 = \begin{bmatrix}z_{0,1}\\z_{0,2}\end{bmatrix}$ is the vector recording Ed and Steve's corresponding earnings, respectively, then the equilibrium mixed strategies must satisfy the following conditions:

$$z_{0,1} = \max_{\mathbf{x}} \left\{ \mathbf{y}_0^t A\mathbf{x} \mid \mathbf{e}^t\mathbf{x} = 1 \text{ and } \mathbf{x} \geq \mathbf{0} \right\} \tag{8.45}$$

and

$$z_{0,2} = \max_{\mathbf{y}} \left\{ \mathbf{y}^t B\mathbf{x}_0 \mid \mathbf{e}^t\mathbf{y} = 1 \text{ and } \mathbf{y} \geq \mathbf{0} \right\}.$$

Observe that due to the manner in which equilibrium strategies, $\mathbf{x}_0$ and $\mathbf{y}_0$, are defined, if one player deviates from his equilibrium strategy while the second player continues to follow his own, then the first sees no improvement in his earnings.

Finding the vector quantities, $\mathbf{x}_0$, $\mathbf{y}_0$, $\mathbf{z}_0$, that satisfy the conditions in (8.45) can be restated in terms of solving a certain quadratic programming problem. Namely, the triple, $\begin{bmatrix}\mathbf{z}_0\\\mathbf{x}_0\\\mathbf{y}_0\end{bmatrix}$, constitutes a solution of (8.45) if and only if it is also

a solution of the following:

$$\min_{\mathbf{z},\mathbf{x},\mathbf{y}} f(\mathbf{z},\mathbf{x},\mathbf{y}) = \mathbf{e}^t\mathbf{z} - \mathbf{x}^t(A+B)\mathbf{y} \tag{8.46}$$

subject to

$$\begin{aligned} A^t\mathbf{y} &\leq z_1\mathbf{e} \\ B\mathbf{x} &\leq z_2\mathbf{e} \\ \mathbf{e}^t\mathbf{x} &= 1 \\ \mathbf{e}^t\mathbf{y} &= 1 \\ \mathbf{x},\mathbf{y} &\geq \mathbf{0}, \end{aligned}$$

where $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$. The proof that the solution of (8.46) coincides with that of (8.45) is beyond the scope of this text and can be found in other sources [27]. We will focus on the connection between this problem and that of the general quadratic programming model and apply these tools to Ed and Steve's game. We will also leave it as an exercise to show that when $B = -A$, the solutions of (8.45) and (8.46) are not only identical but also coincide with the solution of the zero-sum matrix game.

We first express (8.46) in the standard form from (8.35). For the sake of compact notation, we let $\mathbf{w}$ be the vector in $\mathbb{R}^6$, whose first two entries we associate to $\mathbf{z}$, the next two to $\mathbf{x}$, and the last two to $\mathbf{y}$. In other words, $\mathbf{w} = \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{y} \end{bmatrix}$.

Now define $\mathbf{p} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ and

$$Q = \begin{bmatrix} 0_{2\times 2} & 0_{2\times 2} & 0_{2\times 2} \\ 0_{2\times 2} & 0_{2\times 2} & -(A+B) \\ 0_{2\times 2} & -(A+B)^t & 0_{2\times 2} \end{bmatrix}$$

Straightforward calculations show that the objective function, f, from (8.46) is none other than the quadratic function $f : \mathbb{R}^6 \to \mathbb{R}$ given by

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^tQ\mathbf{w} + \mathbf{p}^t\mathbf{w}. \tag{8.47}$$

We now seek to express the constraints from (8.46) in the form of a matrix

inequality, $C\mathbf{w} \leq \mathbf{d}$, along with a matrix equation $E\mathbf{w} = \mathbf{b}$. (We use E instead of A since A already denotes Ed's payoff matrix.)

Equality constraints from (8.46) dictate $\mathbf{e}^t\mathbf{x} = 1$ and $\mathbf{e}^t\mathbf{y} = 1$, so we set $\mathbf{b} = \mathbf{e}$ and $E = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$. The matrix C has 8 rows and 6 columns. Four of the rows correspond to the sign restrictions $-\mathbf{x} \leq \mathbf{0}$ and $-\mathbf{y} \leq \mathbf{0}$. The remaining four arise from the inequalities, $A^t\mathbf{y} \leq z_1\mathbf{e}$ and $B\mathbf{x} \leq z_2\mathbf{e}$, which can be rewritten as $-z_1\mathbf{e} + A^t\mathbf{y} \leq \mathbf{0}$ and $-z_2\mathbf{e} + B\mathbf{x} \leq \mathbf{0}$. Set $\mathbf{d} = \mathbf{0}_{8\times 1}$, $M_1 = \begin{bmatrix} -1 & 0 \\ -1 & 0 \end{bmatrix}$, and $M_2 = \begin{bmatrix} 0 & -1 \\ 0 & -1 \end{bmatrix}$. Then inequality constraints are summarized through the inequality $C\mathbf{x} \leq \mathbf{d}$, where

$$C = \begin{bmatrix} M_1 & 0_{2\times 2} & A^t \\ M_2 & B & 0_{2\times 2} \\ 0_{2\times 2} & -I_2 & 0_{2\times 2} \\ 0_{2\times 2} & 0_{2\times 2} & -I_2 \end{bmatrix}.$$

Knowing Q, E, $\mathbf{b}$, and $\mathbf{d}$, we now restate (8.46) in the standard quadratic program form:

$$\begin{aligned} &\text{minimize } f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^tQ\mathbf{w} + \mathbf{p}^t\mathbf{w} \\ &\text{subject to} \\ &\qquad E\mathbf{w} = \mathbf{b} \\ &\qquad C\mathbf{w} \leq \mathbf{d}. \end{aligned} \tag{8.48}$$

Waypoint 8.4.1. Use Maple's `QPSolve` command to verify that the solution of (8.48) is given by $\mathbf{w}_0 = \begin{bmatrix} 5/3 \\ 7/2 \\ 1/2 \\ 1/2 \\ 2/3 \\ 1/3 \end{bmatrix}$. Thus Ed's equilibrium strategy consists of $\mathbf{x}_0 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$, implying he chooses each column with equal probability. His earnings are given by $z_{0,1} = 5/3$. Steve, on the other hand, has an equilibrium strategy of $\mathbf{y}_0 = \begin{bmatrix} 2/3 \\ 1/3 \end{bmatrix}$, meaning he chooses the first row 2/3 of the time and has earnings of $z_{0,2} = 7/2$. That Steve's earnings are much larger than Ed's is intuitively obvious if we compare the relative sizes of the entries in A and B.

We can verify these results by applying our newly developed tools for quadratic programming problems. Using tools for calculating KKT points, as discussed in Section 8.2, we can verify $\mathbf{w}_0$ is the only KKT point of (8.48). The multiplier vector, $\boldsymbol{\lambda}_0$, corresponding to the eight inequality constraints is given by $\boldsymbol{\lambda}_0 = \begin{bmatrix} 1/6 \\ 5/6 \\ 1/6 \\ 5/6 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, and that the multiplier vector corresponding to the two equality constraints is $\boldsymbol{\mu}_0 = \begin{bmatrix} 11/6 \\ 11/3 \end{bmatrix}$. Only the first four constraints of to $C\mathbf{w} \leq \mathbf{d}$ are binding. Thus, to apply the bordered Hessian test, we form a submatrix of C using its top four rows:

$$\tilde{C} = \begin{bmatrix} M_1 & 0_{2\times 2} & A^t \\ M_2 & B & 0_{2\times 2} \end{bmatrix}.$$

It is easy to check that the rows forming $\begin{bmatrix} E \\ \tilde{C} \end{bmatrix}$ are linearly independent, so $\mathbf{x}_0$ is regular.

The bordered Hessian is the 12-by-12 matrix given by

$$B_h = \begin{bmatrix} 0_{2\times 2} & 0_{2\times 4} & E \\ 0_{4\times 2} & 0_{4\times 4} & \tilde{C} \\ E^t & \tilde{C}^t & Q \end{bmatrix}.$$

In this case $n = 6$ (the number of decision variables), $m = 2$ (the number of equality constraints), and $k = 4$ (the number of binding inequality constraints). Since $n - (m + k) = 0$, we need only to check that $\det(B)$ has the same sign as $(-1)^{m+k} = 1$. A quick calculation verifies that $\det(B_h) = 36$ so that the KKT point $\mathbf{w}_0$ is a solution of (8.48).

We end this section by noting that the mixed strategy Nash equilibrium solution of a bimatrix game need not be unique in general. It can be the case that (8.46) has multiple solutions. While the value of the objective function, f, in (8.46) must be the same for all of these, the players' equilibrium mixed strategies and corresponding payoffs, as determined by entries of $\mathbf{x}_0$, $\mathbf{y}_0$, and $\mathbf{z}_0$, may vary.

Exercises Section 8.4

1. Consider the quadratic programming problem given by

$$\begin{aligned} \text{minimize} & f(x_1, x_2) = 5x_1^2 + 4x_1x_3 + 8x_2^2 + x_3^2 + x1 + 2x_2 - x_3 \\ & \text{subject to} \\ & x_1 - x_2 + x_3 = 1 \\ & x_1 + 2x_2 + x_3 = -3. \end{aligned}$$

   (a) Rewrite the problem in the standard form (8.35). (Hint: The matrix Q is simply the Hessian of f.)

   (b) Explain why the problem is convex. Then determine its solution using formula (8.39).

2. Solve each of the following nonconvex, quadratic programming problems.

   (a)

$$\begin{aligned} \text{minimize } & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x} + \mathbf{p}^t\mathbf{x} \\ & \text{subject to} \\ & A\mathbf{x} = \mathbf{b}, \end{aligned}$$

where $Q = \begin{bmatrix} 1 & -2 & 0 \\ -2 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 2 & -1 \\ 4 & 0 & 1 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

(b)

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x} \\ &\text{subject to} \\ &C\mathbf{x} \leq \mathbf{d}, \end{aligned}$$

where $Q = \begin{bmatrix} 1 & -2 & 0 \\ -2 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 2 & 2 \\ -1 & -1 & -3 \\ 1 & 1 & 0 \end{bmatrix}$, and $\mathbf{d} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$.

(c)

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t Q\mathbf{x} + \mathbf{p}^t\mathbf{x} \\ &\text{subject to} \\ &A\mathbf{x} = \mathbf{b} \\ &C\mathbf{x} \leq \mathbf{d}, \end{aligned}$$

where $Q = \begin{bmatrix} 2 & 4 & 0 & 0 \\ 4 & 7 & -2 & -1 \\ 0 & -2 & -3 & -1 \\ 0 & -1 & -1 & 0 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 2 \\ -3 \\ 0 \\ 1 \end{bmatrix}$. $A = \begin{bmatrix} 1 & 2 & 3 & -1 \\ 0 & -5 & 4 & 2 \\ 3 & 1 & 5 & 6 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 1 & 1 & 3 \\ -1 & 3 & 1 & 0 \end{bmatrix}$ and $\mathbf{d} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

3. Verify formula (8.49). That is, show that

$$\begin{bmatrix} 0_{m\times m} & A \\ A^t & Q \end{bmatrix}^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}AQ^{-1} \\ -Q^{-1}A^tS^{-1} & Q^{-1}(Q + A^tS^{-1}A)Q^{-1} \end{bmatrix}, \tag{8.49}$$

where $S = -AQ^{-1}A^t$.

4. An individual's blood type (A, B, AB, or O) is determined by a pair of inherited alleles, of which there are three possibilities: A, B, and O, where A and B are dominant over O. Table 8.1 summarizes the blood types produced by each allele pair. Note that the allele pairs AO=OA and BO=OB result in blood types A and B, respectively, due to the dominate nature of A and B.

Let $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ be a vector whose entries represent the frequencies of

TABLE 8.1: Blood types produced by different allele pairs

	First Allele		
Second	A	B	O
A	AA=A	AB	AO=A
B	BA=AB	BB=B	BO=B
O	OA=A	OB=B	OO=O

the alleles, A, B, and C, within a certain population. An individual is heterozygous for blood type if he or she inherits different allele types. Since

$$1 = (x_1 + x_2 + x_3)^2 = x_1^2 + 2x_1x_2 + 2x_1x_3 + x_2^2 + 2x_2x_3 + x_3^2,$$

the probability that an individual is heterozygous is then given by $2x_1x_2 + 2x_1x_3 + 2x_2x_3$.

Determine the values of x_1, x_2, and x_3 that maximize the preceding probability by constructing and then solving a quadratic programming problem of the form (8.35).

5. Consider the special case of the bimatrix game when Steve's payoff matrix, B, equals $-A$. Show that if we identify z_1 with w and z_2 with $-z$, then the solution of (8.46) coincides with that of the zero-sum matrix game as formulated in Section 4.1.5. (Hint: When $B = -A$, the objective in (8.46) reduces to one of minimizing

$$\begin{aligned} f(\mathbf{z}, \mathbf{x}, \mathbf{y}) &= \mathbf{e}^t\mathbf{z} - \mathbf{x}^t(A + B)\mathbf{y} \\ &= \mathbf{e}^t\mathbf{z} \\ &= z_1 + z_2 \\ &= w - z, \end{aligned}$$

which is achieved by minimizing w and maximizing z. Show the constraints from (8.46) then coincide with the feasibility of $\mathbf{x}$ for Ed's (primal) LP and $\mathbf{y}$ for Steve's (dual) LP as constructed in Section 4.1.5.)

6. In the game of "chicken" two individuals drive their cars toward one another at high speeds. If Driver A swerves and Driver B continues along his or her straight-line path, then Driver B receives recognition for his bravery and Driver A is deemed a "chicken." To these outcomes we associate payoffs of 2 and 0, respectively. Similarly, if Driver B swerves and Driver A continues along his straight-line path, then Driver A is considered "brave" and Driver B a "chicken." If both drivers continue straight-on, they crash into one another, an outcome we associate with

payoffs of -1 for each driver. Finally, if both drivers swerve, they are each awarded payoffs of 1 for having the common sense to avoid a collision.

Based upon the rules of the game, we can associate to the two drivers the following payoff matrices, where row 1 and column 1 are associated with continuing straight and row 2 and column 2 to swerving:

$$A = \begin{bmatrix} -1 & 0 \\ 2 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} -1 & 2 \\ 0 & 1 \end{bmatrix}$$

Set up and solve a quadratic programming that determines the all Nash equilibria for the game. There are three. Two are pure strategy equilibria, where one driver always continues driving straight and the other swerves, and vice versa. The third equilibrium is of mixed-strategy type in which each driver elects to continue straight or to swerve with a non-zero probability.

8.5 Sequential Quadratic Programming

Sequential Quadratic Programming is a powerful, iterative technique for solving nonlinear programming problems, well-suited for those whose constraints are nonlinear.

8.5.1 Method Derivation for Equality-type Constraints

An important connection exists between Newton's method and sequential quadratic programming. To best understand the nature of this connection, we first develop the technique in the case of equality-type constraints. Thus, we focus on the problem

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to} \\ &\quad h_1(\mathbf{x}) = 0 \\ &\quad h_2(\mathbf{x}) = 0 \\ &\quad \vdots \\ &\quad h_p(\mathbf{x}) = 0, \end{aligned} \tag{8.50}$$

where f and each h_j, $1 \leq j \leq p$, is a twice-differentiable function having domain $S \subseteq \mathbb{R}^n$. In compact form, (8.50) becomes

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to} \\ &\quad \mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned} \tag{8.51}$$

where $\mathbf{h} : S \to \mathbb{R}^p$ is the vector-valued function defined by $\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_p(\mathbf{x}) \end{bmatrix}$.

In a nutshell, to solve (8.51), we will apply Newton's Method to the associated Lagrangian function. At each iteration, we solve a quadratic programming problem. The matrix associated with the objective function is the Hessian of the Lagrangian. Constraints are formulated using the gradient of L, the Jacobian of $\mathbf{h}$, and $\mathbf{h}$ itself.

The Lagrangian function, L, associated with (8.51) is given by

$$L(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}), \tag{8.52}$$

where $\mathbf{x} \in S$ and $\boldsymbol{\mu} \in \mathbb{R}^p$.

Recall results (8.29) and (8.29) from Section 8.3. These demonstrated that if $\mathbf{x}_0$ is a KKT point of NLP (8.51) with corresponding multiplier vector, $\boldsymbol{\mu}_0$, then the gradient of L with respect to both vector variables, $\mathbf{x}$ and $\boldsymbol{\mu}$, vanishes at $(\mathbf{x}_0, \boldsymbol{\mu}_0)$. In other words, $(\mathbf{x}_0, \boldsymbol{\mu}_0)$ is a critical point of L.

The nature of this critical point provides information useful for solving (8.51). If this critical point is a minimum of L on $S \times \mathbb{R}^p$, then for all feasible $\mathbf{x}$ in S,

$$\begin{aligned} f(\mathbf{x}_0) &= f(\mathbf{x}_0) + \boldsymbol{\mu}_0^t \mathbf{h}(\mathbf{x}_0) \\ &= L(\mathbf{x}_0, \boldsymbol{\mu}_0) \\ &\leq L(\mathbf{x}, \boldsymbol{\mu}_0) \\ &= f(\mathbf{x}) + \boldsymbol{\mu}_0^t \mathbf{h}(\mathbf{x}) \\ &= f(\mathbf{x}). \end{aligned} \tag{8.53}$$

Hence, $\mathbf{x}_0$ is a solution of (8.51). For many examples, such as the *ConPro Manufacturing Company*NLP, $(\mathbf{x}_0, \boldsymbol{\mu}_0)$ is not a minimum of L, but instead corresponds to a saddle point, as defined in Section 8.3. Fortunately, Theorem 8.3.1 tells us that this is also a sufficient condition for $\mathbf{x}_0$ to be a solution of (8.51).

One means of estimating the critical point, $(\mathbf{x}_0, \boldsymbol{\mu}_0)$, is to apply Newton's Method to L. Suppose $\mathbf{w}_1 = \begin{bmatrix} \mathbf{x}_1 \\ \boldsymbol{\mu}_1 \end{bmatrix}$ is our initial value, where $\mathbf{x}_1$ belongs to S and and $\boldsymbol{\mu}_1$ belongs to $\mathbb{R}^p$. Denote the Newton direction of L at $\mathbf{w}_1$ by

$$\Delta \mathbf{w} = \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\mu} \end{bmatrix}. \tag{8.54}$$

Determining $\Delta \mathbf{w}$ requires evaluating at $\mathbf{w}_1$ the gradient and Hessian of L in both vector variables, $\mathbf{x}$ and $\boldsymbol{\mu}$. Let ∇L and H_L denote these two quantities, where rows and columns are labeled so that differentiation takes place with respect to the components of $\mathbf{x}$ first. Then straightforward calculations establish

$$\nabla L(\mathbf{x}_1, \boldsymbol{\mu}_1) = \begin{bmatrix} \nabla f(\mathbf{x}_1) + \mathbf{J}_{\mathbf{h}}(\mathbf{x}_1)^t \boldsymbol{\mu}_1 \\ \mathbf{h}(\mathbf{x}_1) \end{bmatrix}, \tag{8.55}$$

and

$$H_L(\mathbf{x}_1, \boldsymbol{\mu}_1) = \begin{bmatrix} H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1) & \mathbf{J}_{\mathbf{h}}(\mathbf{x}_1)^t \\ \mathbf{J}_{\mathbf{h}}(\mathbf{x}_1) & 0_{p \times p} \end{bmatrix}. \tag{8.56}$$

Here we have used $H_{L,\mathbf{x}}$ to denote the Hessian of L in $\mathbf{x}$ alone. In terms of f

and $\mathbf{h}$,

$$H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1) = H_f(\mathbf{x}_1) + \sum_{j=1}^{p} \mu_{1,j} H_{h_j}(\mathbf{x}_1),$$

where $\mu_{1,j}$ denotes the jth component of $\boldsymbol{\mu}_1$.

Using the Newton direction formula, Equation (7.13) from Section 7.2, we obtain

$$-\begin{bmatrix} H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1) & \mathbf{J_h}(\mathbf{x}_1)^t \\ \mathbf{J_h}(\mathbf{x}_1) & 0_{p\times p} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \nabla f(\mathbf{x}_1) + \mathbf{J_h}(\mathbf{x}_1)^t \boldsymbol{\mu}_1 \\ \mathbf{h}(\mathbf{x}_1) \end{bmatrix}. \tag{8.57}$$

Equation (8.57) bears a resemblance to Equation (8.39) from Section 8.4, which arose in the context of quadratic programming. To see this connection more clearly, let us consider the following *quadratic subproblem*:

$$\begin{aligned} &\text{minimize } \frac{1}{2}(\Delta\mathbf{x})^t H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1)\Delta\mathbf{x} + (\nabla f(\mathbf{x}_1) + \mathbf{J_h}(\mathbf{x}_1)^t \boldsymbol{\mu}_1)^t(\Delta\mathbf{x}) \\ &\text{subject to } \mathbf{J_h}(\mathbf{x}_1)(\Delta\mathbf{x}) = -\mathbf{h}(\mathbf{x}_1). \end{aligned} \tag{8.58}$$

For $\Delta\mathbf{x}$ to be a KKT point of (8.58) with associated multiplier vector, $\Delta\boldsymbol{\mu}$, the feasibility condition must be satisfied, i.e., $\mathbf{J_h}(\mathbf{x}_1)(\Delta\mathbf{x}) = -\mathbf{h}(\mathbf{x}_1)$, and the gradient of the associated Lagrangian must vanish, or, in other words,

$$H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1)\Delta\mathbf{x} + (\nabla f(\mathbf{x}_1) + \mathbf{J_h}(\mathbf{x}_1)^t \boldsymbol{\mu}_1) + \mathbf{J_h}(\mathbf{x}_1)^t \Delta\boldsymbol{\mu} = \mathbf{0}.$$

But these conditions combine to yield matrix Equation (8.57).

Thus, the values of $\Delta\mathbf{x}$ and $\Delta\boldsymbol{\mu}$ can be obtained by solving either the quadratic subproblem (8.58), or the matrix equation (8.57). Using these values, we obtain the Newton direction of L at $\mathbf{w}_1 = \begin{bmatrix} \mathbf{x}_1 \\ \boldsymbol{\mu}_1 \end{bmatrix}$. Namely, $\Delta\mathbf{w} = \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\mu} \end{bmatrix}$. Therefore, $\mathbf{w}_2 = \mathbf{w}_1 + \Delta\mathbf{w}$ so that $\mathbf{x}_2 = \mathbf{x}_1 + \Delta\mathbf{x}$ and $\boldsymbol{\mu}_2 = \boldsymbol{\mu}_1 + \Delta\boldsymbol{\mu}$. This completes the first iteration of the *Sequential Quadratic Programming Technique*. We now summarize this process.

The Sequential Programming Technique (SQPT) for NLPs with Equality-type Constraints

To obtain an approximate solution of NLP (8.51) under the assumption that f and each component of $\mathbf{h}$ are twice-differentiable on S:

1. Starting with $\mathbf{x}_1$ in S, along with $\boldsymbol{\mu}_1$ in $\mathbb{R}^p$, use the solution of (8.58), which is also the solution of (8.57), to compute $\Delta\mathbf{x}$ and $\Delta\boldsymbol{\mu}$. Of course, successfully doing so hinges on whether $\begin{bmatrix} H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1) & \mathbf{J_h}(\mathbf{x}_1)^t \\ \mathbf{J_h}(\mathbf{x}_1) & 0_{p\times p} \end{bmatrix}$ is invertible.

2. Set $\mathbf{x}_2 = \mathbf{x}_1 + \Delta\mathbf{x}$ and $\boldsymbol{\mu}_2 = \boldsymbol{\mu}_1 + \Delta\boldsymbol{\mu}$.
3. Return to (1), replace $\mathbf{x}_1$ and $\boldsymbol{\mu}_1$ with $\mathbf{x}_2$ and $\boldsymbol{\mu}_2$, respectively, and repeat the process.
4. Terminate the process when the difference between successive iterates, $\mathbf{x}_{k+1} - \mathbf{x}_k$ is smaller than some specified tolerance.

Under ideal circumstances, the sequence of iterates,

$$\left\{(\mathbf{x}_1, \boldsymbol{\mu}_1), (\mathbf{x}_2, \boldsymbol{\mu}_2), \ldots\right\}$$

converges to $(\mathbf{x}_0, \boldsymbol{\mu}_0)$, where $\mathbf{x}_0$ is a solution of (8.51) having associated multiplier vector, $\boldsymbol{\mu}_0$.

It is important to recognize that the SQPT, for the case of equality constraints. generates a sequence of iterates identical to that obtained by applying Newton's Method to the associated Lagrangian function.

To demonstrate this new technique, we consider the NLP

$$\begin{aligned} &\text{minimize} f(x_1, x_2) = \sin(x_1 - x_2) \\ &\text{subject to} \\ &x_1^2 + x_2^2 = 1 \end{aligned} \tag{8.59}$$

The solution of (8.59) can of course be obtained by solving the constraint for one decision variable, substituting the result into the objective, and minimizing the resulting function of a single variable. Doing so leads to a solution of $\mathbf{x}_0 = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ with corresponding objective value $f(\mathbf{x}_0) = -\sin(\sqrt{2})$.

While NLP (8.59) is a very simple example, it is also well suited for demonstrating our new technique, due to the nature of the objective function and the nonlinear constraint.

We use as our initial value $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\boldsymbol{\mu}_1 = 1$. (For the sake of consistency with our previous derivations, we express $\boldsymbol{\mu}_1$ and $\mathbf{h}$ using vector notation even though both are scalar-valued for this particular problem.) The quantities needed to compute $\Delta\mathbf{x}$ and $\Delta\boldsymbol{\mu}$ consist of the following:

1. $\mathbf{h}(\mathbf{x}) = x_1^2 + x_2^2 - 1$
2. $\nabla f(\mathbf{x}) = \begin{bmatrix} \cos(x_1 - x_2) \\ -\cos(x_1 - x_2) \end{bmatrix}$

3. $\mathbf{J_h}(\mathbf{x}) = \begin{bmatrix} 2x_1 & 2x_2 \end{bmatrix}$

4. $L(\mathbf{x}, \boldsymbol{\mu}) = \sin(x_1 - x_2) + \boldsymbol{\mu}(x_1^2 + x_2^2 - 1)$

5. $H_{L,\mathbf{x}}(\mathbf{x}, \boldsymbol{\mu}) = \begin{bmatrix} -\sin(x_1 - x_2) + 2\boldsymbol{\mu} & \sin(x_1 - x_2) \\ \sin(x_1 - x_2) & -\sin(x_1 - x_2) + 2\boldsymbol{\mu} \end{bmatrix}$

Evaluation of these quantities at $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\boldsymbol{\mu}_1 = 1$ and substitution of the results into (8.57) yields the following matrix equation:

$$-\begin{bmatrix} \sin(1) + 2 & -\sin(1) & 2 \\ -\sin(1) & \sin(1) + 2 & 4 \\ 2 & 4 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \cos(1) + 2 \\ -\cos(1) + 4 \\ 4 \end{bmatrix}. \tag{8.60}$$

Solving (8.60), we obtain $\begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\mu} \end{bmatrix} \approx \begin{bmatrix} -.7 \\ -.650 \\ -.550 \end{bmatrix}$, from which it follows that

$$\begin{aligned} \mathbf{x}_2 &\approx \mathbf{x}_1 + \Delta\mathbf{x} \\ &= \begin{bmatrix} .3 \\ 1.35 \end{bmatrix} \end{aligned}$$

and $\boldsymbol{\mu}_2 = \boldsymbol{\mu}_1 + \Delta\boldsymbol{\mu} \approx .45$.

Table 8.2 provides values of $\mathbf{x}_k$ and $\boldsymbol{\mu}_k$ for the first seven iterations of the SQPT applied to NLP (8.59) using $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\boldsymbol{\mu}_1 = 1$.

TABLE 8.2: Results of the Sequential Quadratic Programming Technique applied to NLP (8.59)

k	$\mathbf{x}_k^t$	$\boldsymbol{\mu}_k$
1	[1, 2]	1
2	[.3, 1.35]	.45
3	[−.161, 1.115]	1.90
4	[−.474, .949]	.095
5	[−.678, .780]	.075
6	[−.730, .691]	.104
7	[−.706, .708]	.110
8	[−.707, .707]	.110

The values of $\mathbf{x}_k$ clearly converge to the solution of (8.59), namely $\mathbf{x}_0 = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$. But to what do the values of $\boldsymbol{\mu}_k$ converge? The answer of course is the corresponding Lagrange multiplier.

Waypoint 8.5.1. Verify that $\mathbf{x}_0 = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ is a KKT point of (8.59) with corresponding multiplier, $\mu_0 = \dfrac{\cos(\sqrt{2})}{\sqrt{2}} \approx .110$. Then use the `NewtonsMethod` procedure from Section 7.2.3 to verify that the values in Table 8.2 coincide with those obtained by applying Newton's Method to the Lagrangian, L, using an initial value $\begin{bmatrix} \mathbf{x}_1 \\ \mu_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$. Note that the procedure must be modified slightly to reflect the fact the Lagrangian is a function of three variables, as opposed to two. The three consist of the two components of $\mathbf{x}$, along with μ.

8.5.2 The Convergence Issue

The SQPT is based upon Newton's Method, which, as we discovered in Section 7.2.4, exhibits a quadratic rate of convergence to the desired minimum, under appropriate conditions. Table 8.2 exhibits this fast rate of convergence. At the same time, a numeric technique such as Newton's Method can generate a sequence of iterates converging to a critical point that is not a minimum.

It should not come as a surprise then that a sequence of SQPT iterates, $\{(\mathbf{x}_1, \mu_1), (\mathbf{x}_2, \mu_2), \ldots\}$, can have the same undesirable property. We can demonstrate this phenomenon by applying the SQPT to NLP (8.59) using a different initial value. If $\mathbf{x}_1 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$ and $\mu_1 = -1$, then the resulting sequences of iterates do not converge to $\mathbf{x}_0 = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ and $\mu_0 = \dfrac{\cos(\sqrt{2})}{\sqrt{2}}$, respectively. Instead, the sequence $\{\mathbf{x}_k\}$ converges to the second KKT point of NLP (8.59), $\mathbf{x}_0 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$, and the sequence $\{\mu_k\}$ converges to the corresponding multiplier vector, $\mu_0 = -\dfrac{\cos(\sqrt{2})}{\sqrt{2}} \approx -.110$. Note that while $\mathbf{x}_0$ in this case is a KKT point, it is not a solution of (8.59). Mere comparison of objective values shows this to be the case, as does an application of the Saddle Point Optimality Test from Section 8.3.2.

8.5.3 Inequality-Type Constraints

In the case where only equality-type constraints are present, solving NLP (8.50) using the SQPT reduces to solving a sequence of quadratic programming problems of the form (8.58), each of which has only equality-type constraints.

If an NLP has inequality-type constraints as well, a modification of the Sequential Programming Technique requires us to solve at each iteration a quadratic programming subproblem where inequality-type constraints are also present. Instead of formally deriving the exact quadratic programming problem as we did in Section 8.5.1, we instead state the end result.

Suppose we add to NLP (8.51) inequality-type constraints, expressing it as

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to} \\ &\qquad \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ &\qquad \mathbf{h}(\mathbf{x}) = \mathbf{0}. \end{aligned} \tag{8.61}$$

Here, $\mathbf{g} : S \to \mathbb{R}^m$ is the vector-valued function each of whose components are twice-differentiable. The Lagrangian, L, is given by

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^t \mathbf{g}(\mathbf{x}) + \boldsymbol{\mu}^t \mathbf{h}(\mathbf{x}),$$

whose Hessian in $\mathbf{x}$ we once again denote by $H_{L,\mathbf{x}}$.

Choose $\mathbf{x}_1$ in S, $\boldsymbol{\lambda}_1$ in $\mathbb{R}^m$ with $\boldsymbol{\lambda}_1 \geq \mathbf{0}$, and $\boldsymbol{\mu}_1$ in $\mathbb{R}^p$. Then set $\mathbf{x}_2 = \mathbf{x}_1 + \Delta\mathbf{x}$, where $\Delta\mathbf{x}$ is the solution the following:

$$\begin{aligned} &\text{minimize } \frac{1}{2}(\Delta\mathbf{x})^t H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\mu}_1)\Delta\mathbf{x} + \nabla f(\mathbf{x}_1)^t \Delta\mathbf{x} \\ &\text{subject to} \\ &\qquad \mathbf{J}_{\mathbf{g}}(\mathbf{x}_1)(\Delta\mathbf{x}) \leq -\mathbf{g}(\mathbf{x}_1) \\ &\qquad \mathbf{J}_{\mathbf{h}}(\mathbf{x}_1)(\Delta\mathbf{x}) = -\mathbf{h}(\mathbf{x}_1). \end{aligned} \tag{8.62}$$

The multiplier vectors $\boldsymbol{\lambda}_2$ and $\boldsymbol{\mu}_2$ are assigned the values associated with the solution of (8.62). Note that this process differs from the situation when only equality-type constraints were present. There, the multiplier vector was labeled $\Delta\boldsymbol{\mu}$, which we then added to $\boldsymbol{\mu}_1$ to obtain $\boldsymbol{\mu}_2$.

For example, consider the 3-variable NLP,

$$\begin{aligned} \text{minimize} & f(x_1, x_2, x_3) = x_1^2 + 2x_1^2x_2x_3 + x_2^2 - x_3^2 \\ & \text{subject to} \\ & x_1 + 2x_2 - x_3 \leq 2 \\ & x_1^2 \qquad\qquad \geq 1 \\ & x_1^2 + x_2^2 + x_3^2 = 4. \end{aligned} \tag{8.63}$$

We will use as initial values, $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, $\boldsymbol{\lambda}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\boldsymbol{\mu}_1 = -1$.

Straightforward calculations lead to the following expressions:

1. $-\mathbf{g}(\mathbf{x}) = -\begin{bmatrix} x_1 + 2x_2 - x_3 - 2 \\ -x_1^2 + 1 \end{bmatrix}$
2. $-\mathbf{h}(\mathbf{x}) = -\left(x_1^2 + x_2^2 + x_3^2 - 4\right)$
3. $\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 + 4x_1x_2x_3 \\ 2x_1^2x_3 + 2x_2 \\ 2x_1^2x_2 - 2x_3 \end{bmatrix}$
4. $\mathbf{J}_{\mathbf{g}}(\mathbf{x}) = \begin{bmatrix} 1 & 2 & -1 \\ 0 & -2x_2 & 0 \end{bmatrix}$
5. $\mathbf{J}_{\mathbf{h}}(\mathbf{x}) = \begin{bmatrix} 2x_1 & 2x_2 & 2x_3 \end{bmatrix}$
6. $$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) &= f(\mathbf{x}) + \boldsymbol{\lambda}^t\mathbf{g}(\mathbf{x}) + \boldsymbol{\mu}^t\mathbf{h}(\mathbf{x}) \\ &= x_1^2 + 2x_1^2x_2x_3 + x_2^2 - x_3^2 + \lambda_1(x_1 + 2x_2 - x_3 - 2) \\ &+ \lambda_2\left(-x_2^2 + 1\right) + \mu_1\left(x_1^2 + x_2^2 + x_3^2 - 4\right) \end{aligned}$$
7. $H_{L,\mathbf{x}}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{bmatrix} 2 + 4x_2x_3 + 2\mu_1 & 4x_1x_3 & 4x_1x_2 \\ 4x_1x_3 & 2 - 2\lambda_2 + 2\mu 1 & 2x_1^2 \\ 4x_1x_2 & 2x_1^2 & -2 + 2\mu_1 \end{bmatrix}$.

Evaluating these quantities at $\mathbf{x}_1$, $\boldsymbol{\lambda}_1$, $\boldsymbol{\mu}_1$, we obtain

1. $-\mathbf{g}(\mathbf{x}_1) = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$
2. $-\mathbf{h}(\mathbf{x}_1) = -10$

3. $\nabla f(\mathbf{x}_1) = \begin{bmatrix} 26 \\ 10 \\ -2 \end{bmatrix}$

4. $\mathbf{J_g}(\mathbf{x}_1) = \begin{bmatrix} 1 & 2 & -1 \\ 0 & -4 & 0 \end{bmatrix}$

5. $\mathbf{J_h}(\mathbf{x}_1) = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}$

6. $H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\lambda}_1, \boldsymbol{\mu}_1) = \begin{bmatrix} 24 & 12 & 8 \\ 12 & -2 & 2 \\ 8 & 2 & -4 \end{bmatrix}$.

Now we substitute these results into (8.62) and solve the resulting quadratic programming problem. To find the KKT point and corresponding multipliers, we could certainly construct the associated Lagrangian and use methods from Section 8.2.1. In an effort to streamline the solution process, however, we will first determine the KKT point using the matrix form of Maple's `QPSolve` command. If we enter the preceding quantities in Maple so that items 1, 2, 3, 4, 5, and 6 correspond to `d`, `b`, `p`, `C`, `A`, and `Q`, respectively, then `QPSolve([p,Q], [C,d,A,b])` returns a value of $\Delta\mathbf{x} \approx \begin{bmatrix} -.4000 \\ -.7500 \\ -1.0335 \end{bmatrix}$.

While the `QPSolve` command does not return the multiplier values, we can obtain them by using $\Delta\mathbf{x}$ along with the given functions. Substituting $\Delta\mathbf{x}$ into each side of the inequality $\mathbf{J_g}(\mathbf{x}_1)(\Delta\mathbf{x}) \leq -\mathbf{g}(\mathbf{x}_1)$ establishes that the constraint corresponding to the second row of $\mathbf{J_g}(\mathbf{x}_1)$ is binding. Thus $\lambda_{2,1}$, the first component of $\boldsymbol{\lambda}_2$, equals zero by complementary slackness.

In the spirit of Section 8.4.2, let us define $\widetilde{C}$ to be the submatrix of $\mathbf{J_g}(\mathbf{x}_1)$ formed using the row corresponding to the second constraint, which is binding. In this example, $\widetilde{C} = \begin{bmatrix} 0 & -4 & 0 \end{bmatrix}$. Moreover, let $\widetilde{\mathbf{d}}$ be the subvector of $\mathbf{g}(\mathbf{x}_1) = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ corresponding to the second constraint, i.e., $\widetilde{\mathbf{d}} = -3$.

Now consider the matrix equation

$$\begin{bmatrix} H_{L,\mathbf{x}}(\mathbf{x}_1, \boldsymbol{\lambda}_1, \boldsymbol{\mu}_1) & \mathbf{J_h}(\mathbf{x}_1)^t & \widetilde{C}^t \\ \mathbf{J_h}(\mathbf{x}_1) & 0 & 0 \\ \widetilde{C} & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \boldsymbol{\mu}_2 \\ \lambda_{2,2} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}_1) \\ -\mathbf{h}(\mathbf{x}_1) \\ -\widetilde{\mathbf{d}} \end{bmatrix}. \tag{8.64}$$

This equation must be satisfied by the triple $\Delta\mathbf{x}, \boldsymbol{\mu}_2, \lambda_{2,2}$ in order for (8.62) to have a solution $\Delta\mathbf{x}$ that yields a non-binding first constraint, i.e., a constraint for which $\lambda_{2,1} = 0$.

Since $\lambda_{2,1} = 0$, we use only the unknown value, $\lambda_{2,2}$ in the vector of unknowns. In this example, substitution of all known quantities into Equation (8.64) yields

$$\begin{bmatrix} 24 & 12 & 8 & 2 & 0 \\ 12 & -2 & 2 & 4 & -4 \\ 8 & 2 & -4 & 6 & 0 \\ 2 & 4 & 6 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \boldsymbol{\mu}_2 \\ \lambda_{2,2} \end{bmatrix} = \begin{bmatrix} -26 \\ -10 \\ 2 \\ -10 \\ 3 \end{bmatrix},$$

from which it follows that $\Delta \mathbf{x} \approx \begin{bmatrix} -.4 \\ -.75 \\ -1.0355 \end{bmatrix}$, $\boldsymbol{\mu}_2 \approx .4268$, and $\lambda_{2,2} \approx 1.587$.

Thus, one iteration of the technique is complete with the results

$$\mathbf{x}_2 = \mathbf{x}_1 + \Delta \mathbf{x} \approx \begin{bmatrix} .6 \\ 1.25 \\ 1.9665 \end{bmatrix}, \boldsymbol{\lambda}_2 \approx \begin{bmatrix} 0 \\ 1.587 \end{bmatrix}, \text{ and } \boldsymbol{\mu}_2 \approx .4268.$$

At this point, we repeat the process, returning to (8.62), replacing $\mathbf{x}_1$, $\boldsymbol{\lambda}_1$, and $\boldsymbol{\mu}_1$, with $\mathbf{x}_1$, $\boldsymbol{\lambda}_1$, and $\boldsymbol{\mu}_1$, respectively, and so on. The results after only the third iteration are given by $\mathbf{x}_4 \approx \begin{bmatrix} .003 \\ 1.00 \\ 1.735 \end{bmatrix}$, $\boldsymbol{\lambda}_4 = \approx \begin{bmatrix} 0 \\ 1.995 \end{bmatrix}$, and $\boldsymbol{\mu}_4 \approx 1.011$. The exact values, which can be determined by finding the KKT point and corresponding multipliers for the original NLP (8.63), are given by $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \\ \sqrt{3} \end{bmatrix}$, $\boldsymbol{\lambda}_0 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, and $\boldsymbol{\mu}_0 = 1$. Thus, a mere three iterations in this case produces quite accurate results.

8.5.4 A Maple Implementation of the Sequential Quadratic Programming Technique

The worksheet **Sequential Quadratic Programming Technique.mw** provides a systematic means by which to execute the SQPT. Here we use it to complete the first iteration of NLP (8.63)

```
> with(VectorCalculus):with(LinearAlgebra):with(Optimization):
> f:=(x1,x2,x3)-> x1^2+2x1^2*x2*x3+x2^2-x3^2:
  # Enter objective function.
> Delf:=unapply(Gradient(f(x1,x2,x3),[x1,x2,x3]),[x1,x2,x3]):
  # Gradient of f.
> h:=(x1,x2,x3)-> <x1^2+x2^2+x3^2-4>:
  # Vector-valued form of equality constraint function.
```

```
> Jh:=unapply(Jacobian(h(x1,x2,x3),[x1,x2,x3]),[x1,x2,x3]):
  # Jacobian function of h.

> g:=(x1,x2,x3)-><x1+2*x2-x3-2,-x^2+1>:
  # Vector-valued form of inequality constraint functions.

> Jg:=unapply(Jacobian(g(x1,x2,x3),[x1,x2,x3]),[x1,x2,x3]):
  # Jacobian function of g.

> lambda:=<lambda1,lambda2>:
  # Create vector of multipliers.

> L:=unapply(f(x1,x2,x3)+Transpose(lambda).g(x1,x2,x3),
  +<mu>.h(x1,x2,x3), [x1,x2,x3,mu,lambda1,lambda2]):
  # Create Lagrangian Function.

> HLx:=unapply(Hessian(L(x1,x2,x3,mu,lambda1,lambda2),[x1,x2,x3]),
  [x1,x2,x3,mu,lambda1,lambda2]):
  # Hessian in x1,x2,x3 of the Lagrangian.

> X1:=1,2,3;mu1:=-1;Lambda1:=1,1;
  # Initial choices for variables and multipliers. Note Lambda1
  is case sensitive since lambda1 has already been defined as a
  variable in the Lagrangian.

> Qsol:=QPSolve([Delf(X1),HLx(X1,mu1,Lambda1)],[Jg(X1),-g(X1),Jh(X1),-h(X1)]);
  # Solve first quadratic subproblem in iteration process.
```

$$Qsol := \left[-8.1547, \begin{bmatrix} -.3994 \\ -.7500 \\ -1.0335 \end{bmatrix}\right]$$

```
> Jg(X1).<Qsol[2][1],Qsol[2][2],Qsol[2][3]>+g(X1);
  # Determine which inequality constraints in solution of subproblem
  are binding.
```

$$(-.8659)\mathbf{e}_x + (0)\mathbf{e}_y$$

```
> ~C:=SubMatrix(Jg(X1),[2],[1,2,3]);
  # The second constraint is binding, so we form a submatrix of
  Jg(X1) using only the second row. Furthermore, we know first
  component of Lambda2 is zero by complementary slackness.
```

$$\sim C := \begin{bmatrix} 0 & -4 & 0 \end{bmatrix}$$

```
> ~d:=SubVector(g(X1),[2]);
  # Use entry of Jg(X1) corresponding to binding constraint.

> B:=<<<HLx(X1,mu1,Lambda1)|Transpose(Jh(X1))|Transpose(~C)>>,
  <Jh(X1)|ZeroMatrix(1,2)>,<~C|ZeroMatrix(1,2)>>;
```

```
# Create coefficient matrix for finding changes in x1, x2, and
x3, along with new multiplier values.
```

$$B := \begin{bmatrix} 24 & 12 & 8 & 2 & 0 \\ 12 & -2 & 2 & 4 & -4 \\ 8 & 2 & -4 & 6 & 0 \\ 2 & 4 & 6 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 \end{bmatrix}$$

```
> w:=evalf(MatrixInverse(B).<-Delf(X1),-h(X1),-~ d>);
```

$$w := \begin{bmatrix} -.3994 \\ -.7500 \\ -1.0335 \\ .4268 \\ 1.5869 \end{bmatrix}$$

```
> X2:=X1[1]+w[1,1],X1[2]+w[2,1],X1[3]+w[3,1];
  # Create X2 using each entry of X1, added to the corresponding
  entry in w.
```

$$X2 := .6006, 1.2500, 1.9665$$

```
> mu2:=w[4,1];
```

$$\mu 2 := .4268$$

```
> Lambda2:=0,w[5,1];
  # Lambda2 has its first entry equal to zero since the first constraint
  in the quadratic subproblem was not binding at the solution.
  We use the fifth entry of w to form the second component of Lambda2.
  Now return to Step 1 and repeat the process again starting at
  X2,mu2,Lambda2.
```

$$\Lambda 2 = 0, 1.5869$$

Subsequent iterations are handled in a similar manner.

8.5.5 An Improved Version of the SQPT

Numerous refinements of the SQPT exist, each intended to improve upon the original method. Perhaps the simplest of these stems from the fact that at each iteration of the SQPT, the value of $\mathbf{x}_{k+1}$ is obtained by adding the entire quantity $\Delta\mathbf{x}$ to $\mathbf{x}_k$. While $\Delta\mathbf{x}$ is the solution of the quadratic subproblem, the value $\mathbf{x}_{k+1}$ may be no less "infeasible" than was $\mathbf{x}_k$ if it was infeasible to start with. Consequently, instead of adding the entire quantity $\Delta\mathbf{x}$ to $\mathbf{x}_k$ in order to obtain $\mathbf{x}_{k+1}$, we should consider adding only a fraction of its value.

The simplest means for carrying out this process is to utilize a decision-making rule, whereby at each iteration, we choose the fraction of $\Delta\mathbf{x}$ that best

decreases the objective value while simultaneously minimizing the "infeasibility error." This task is accomplished by introducing a new function,

$$M(\mathbf{x}) = f(\mathbf{x}) + \rho P(\mathbf{x}), \tag{8.65}$$

where P is a *penalty function* that measures the extent of constraint violation and where ρ is a fixed, large positive real number. The function, M, is referred to as a *merit function*. Corresponding to NLP (8.61), various different choices exist for P, with one of the most common being

$$P(\mathbf{x}) = \sum_{i=1}^{m} \max(g_i(\mathbf{x}), 0)^2 + \sum_{j=1}^{p} h_j(\mathbf{x})^2. \tag{8.66}$$

Here, *max* denotes the maximum function,

$$\max(x, 0) = \begin{cases} x, & \text{if } x > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{8.67}$$

Note that if $\mathbf{x}$ is feasible for (8.61), then $P(\mathbf{x}) = 0$, and that the value of P increases with the overall amount of constraint violation.

With this choice of P, the merit function becomes

$$M(\mathbf{x}) = f(\mathbf{x}) + \rho \left(\sum_{i=1}^{m} \max(g_i(\mathbf{x}), 0)^2 + \sum_{j=1}^{p} h_j(\mathbf{x})^2 \right). \tag{8.68}$$

Our decision-making rule is to scale $\Delta\mathbf{x}$ at each iteration by an amount, t_0, where t_0 is the smallest positive local minimum of $\phi(t) = M(\mathbf{x}_k + t\Delta\mathbf{x})$. Then we set $\mathbf{x}_{k+1} = \mathbf{x}_k + t_0\Delta\mathbf{x}$. However, we continue to determine $\boldsymbol{\lambda}_{k+1}$ and $\boldsymbol{\mu}_{k+1}$ as we did for the original technique. The new method based upon an appropriate scaling of $\Delta\mathbf{x}$, is known as the *Merit Function Sequential Programming Technique*, or MSQPT.

To illustrate how this modification is an improvement upon the original technique, we revisit NLP (8.63). Recall that $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and that at the end of the first iteration, $\Delta\mathbf{x} \approx \begin{bmatrix} -.4 \\ -.75 \\ -1.034 \end{bmatrix}$. Using $\rho = 10$, the function ϕ becomes

$$\begin{aligned} \phi(t) &= M(\mathbf{x}_1 + t\Delta\mathbf{x}) \\ &= f(1 - .4t, 2 - .75t, 3 - 1.034t) \\ &\quad + 10\left(\sum_{i=1}^{2} \max(g_i(1 - .4t, 2 - .75t, 3 - 1.034t), 0)^2 + h(1 - .4t, 2 - .75t, 3 - 1.034t)^2 \right). \end{aligned} \tag{8.69}$$

Due to the piecewise nature of P, the formula for ϕ is quite complicated. However, its graph, shown in Figure (8.3), clearly indicates the desired positive local minimum.

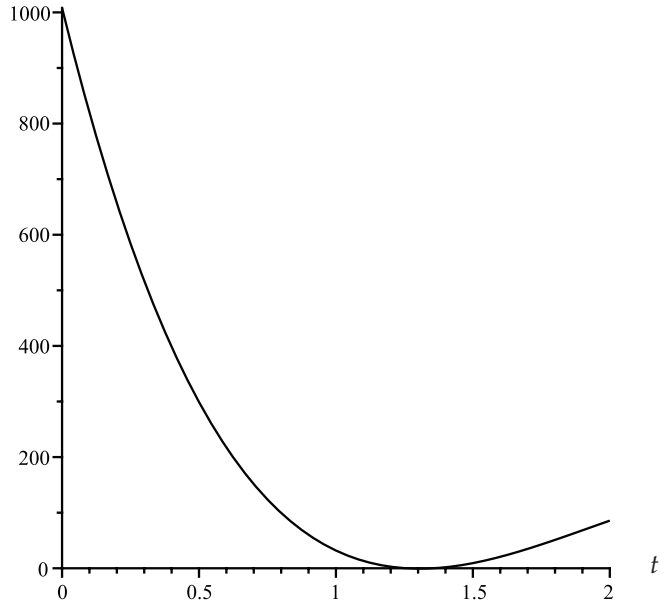


FIGURE 8.3: Graph of $\phi(t) = M(\mathbf{x}_1 + t\Delta\mathbf{x})$.

A single-variable implementation of Newton's Method demonstrates that the minimum occurs at $t_0 \approx 1.3062$. Thus,

$$\begin{aligned} \mathbf{x}_2 &= \mathbf{x}_1 + t_0\Delta\mathbf{x} \\ &\approx \begin{bmatrix}1\\2\\3\end{bmatrix} + 1.3062\begin{bmatrix}-.4\\-.75\\-1.034\end{bmatrix} \\ &\approx \begin{bmatrix}.478\\1.020\\1.649\end{bmatrix}. \end{aligned} \tag{8.70}$$

Comparing this value to that obtained using the SQPT, given by

$\mathbf{x}_2 = \mathbf{x}_1 + \Delta\mathbf{x} \approx \begin{bmatrix} .6 \\ 1.25 \\ 1.966 \end{bmatrix}$, we see that the MSQPT yields a first iterate significantly closer to the NLP's solution of $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \\ \sqrt{3} \end{bmatrix}$.

The Maple worksheet, **Sequential Quadratic Programming Technique.mw**, is easily modified to incorporate use of the merit function. After the objective and constraint functions, `f`, `g`, and `h` have been defined, the penalty and merit functions are constructed using the following input:

```
> P:=unapply(add(piecewise(g(x1,x2,x3)>=0,g(x1,x2,x3)[i]^2,i=1..2)
  +h(x1,x2,x3)^2,[x1,x2,x3])):
> M:=unapply(f(x1,x2,x3)+10*P(x1,x2,x3),[x1,x2,x3]):
```

To account for the scaling of $\Delta\mathbf{x}$, we modify the worksheet after the computation of `w` as follows:

```
> w:=evalf(MatrixInverse(B).<-Delf(X1),-h(X1),-(~ d)>);
```

$$w := \begin{bmatrix} -.3994 \\ -.7500 \\ -1.0335 \\ .4268 \\ 1.5869 \end{bmatrix}$$

```
> phi:=t->M(X1[1]+t*w[1,1],X1[2]+t*w[2,1],X1[3]+t*w[3,1]):
> t0:=NewtonsMethod(phi(t),1,5,.01);
  # NewtonsMethod procedure modified for a function of one variable,
  using an initial value 1, maximum number of five iterations,
  and tolerance of .01.
```

$$t0 := 1.3062$$

```
> X2:=X1[1]+t0*w[1,1],X1[2]+t0*w[2,1],X1[3]+to*w[3,1];
```

$$.478 \quad 1.020 \quad 1.649$$

Subsequent iterations to are handled in a similar manner.

Exercises Section 8.5

1. For each of the following NLPs, estimate the solution by performing three iterations of the SQPT using the provided initial values

(a)

$$\begin{aligned} \text{minimize} & f(x_1, x_2) = x_1^2 + 2x_1x_2x_3 - x_3^2 \\ & \text{subject to} \\ & x_1^2 + x_2^2 + x_3^2 = 10 \\ & 2x_1 + x_2 - x_3 = 1 \end{aligned}$$

along with $\mathbf{x}_1 = \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}$ and $\boldsymbol{\mu}_1 = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$. (Verify that your sequence of values, $\{(\mathbf{x}_1, \boldsymbol{\mu}_1), (\mathbf{x}_2, \boldsymbol{\mu}_2), \ldots\}$, agrees with that obtained by applying Newton's Method to the associated Lagrangian function.)

(b)

$$\begin{aligned} \text{minimize} & f(x_1, x_2) = x_1^2 + 2x_1 - x_1x_2 - 3x_2^2 \\ & \text{subject to} \\ & x_1 - \sin(x_2) \leq 0 \\ & 2x_1^2 + x_2 \leq 1 \\ & x_2 \geq 0 \end{aligned}$$

along with $\mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\boldsymbol{\lambda}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\mu_1 = 1$.

Consider the NLP

$$\begin{aligned} \text{minimize} & f(x_1, x_2) = -x_1^2 + x_2^2 \\ & \text{subject to} \\ & 2x_1^2 + x_2^2 = 6. \end{aligned}$$

Solve this NLP and demonstrate, by choosing different initial values, how it is possible to generate a sequence of SQPT iterates that converges to a KKT point that is not the solution.

2. Use the SQPT to estimate the solution of the *ConPro Manufacturing Company*NLP, (6.6).

3. Recall *Pam's Pentathlon Training Program* NLP, as discussed in Exercise 2, from Section 6.1.

$$\begin{aligned}\text{maximize } f(x_1, x_2, x_3) &= .11193(254 - (180 - .5x_1 - x_2 - x_3))^{1.88} \\ &\quad + 56.0211(5.2 + .1x_1)^{1.05} \\ \text{subject to} \\ &6 \leq x_1 + x_2 + x_3 \leq 10 \\ &2 \leq x_1 \leq 4 \\ &3 \leq x_2 \leq 4 \\ &.6x_1 - .4x_3 \leq 0 \\ &x_1, x_2, x_3 \geq 0.\end{aligned}$$

The decision variables, x_1, x_2, and x_3, denote the total number hours per week Pam devotes to weight lifting, distance running, and speed workouts, respectively. The objective function represents the portion of Pam's total pentathlon score stemming from her performances in the 800 meter run and shot put. It is based upon International Amateur Athletic Federation scoring formulas, together with the fact that Pam currently completes the 800 meter run in 3 minutes and throws the shot put 6.7 meters. Including sign restrictions, the NLP has ten constraints, which reflect requirements as to how Pam allots her training time.

Use the MSQPT, with initial values $\mathbf{x}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ and $\boldsymbol{\lambda}_1 = .5\mathbf{e}$, where $\mathbf{e}$ is the vector in $\mathbb{R}^{10}$ all of whose entries are one, to estimate the solution of this NLP.

Appendix A

Projects

A.1 Excavating and Leveling a Large Land Tract

An excavation company must prepare a large tract of land for future construction.[1] The company recognizes the difficulty and lack of aesthetics associated with leveling the site to form one horizontal plane. Instead, it divides the site into eight rectangles, with each rectangle assigned to fall in one of three possible planes, whose equations are to be determined. Rectangles within a given plane must be pairwise adjacent.

Each plane will be expressed as a function of x and y, and to keep the planes relatively "flat," we require that the slopes in both the x- and y-directions are close to zero. (The actual tolerance will be prescribed as a constraint in the LP.) In addition, adjacent planes should meet along their edges so as to avoid "jumps." This entire process of plane-fitting and leveling is accomplished through a process of transporting among the various rectangles. The company's goal is to minimize the total costs stemming from the digging of dirt, the filling of dirt, and the transporting of dirt from one rectangle to another.

The eight rectangles and their designated planes are shown in Figure A.1.

Corresponding to rectangle i, $1 \leq i \leq 8$, is its center (x_i, y_i) in $\mathbb{R}^2$, its area, A_i, and its average height, h_i. The excavation is comprised of three different types of processes: "cutting," or digging dirt from a rectangle, which decreases the rectangle's average height; transporting dirt from one rectangle to another; and, finally, "filling," or adding dirt to a rectangle, which increases the rectangle's average height.

We define the decision variables for this model using the following notation:

- c_i, where $1 \leq i \leq 8$: amount of "cut" from rectangle i, measured in feet.
- f_i, $1 \leq i \leq 8$: amount of "fill" for rectangle i, measured in feet.
- t_{ij}, where $1 \leq i \leq 8$ and $1 \leq j \leq 8$: amount of dirt transported from rectangle i to rectangle j, measured in units of cubic feet, where we assume $t_{ii} = 0$ for $i = 1, 2, \ldots, 8$.
- a_j, $1 \leq j \leq 3$: the slope in the x-direction of plane j.
- b_j, $1 \leq j \leq 3$: the slope in the y-direction of plane j.
- d_j, $1 \leq j \leq 3$: the vertical axis intercept of plane j.

[1] Based upon Moreb and Bafail, [33], (1994).

To create an LP with nonnegative decision variables, we rewrite all slopes and intercepts of the planes as differences of two nonnegative decision variables:

$$a_j = a_{j1} - a_{j2},\ b_j = b_{j1} - b_{j2}, \text{ and } d_j = d_{j1} - d_{j2}, \quad 1 \le j \le 3,$$

where a_{jk} b_{jk}, and d_{jk} is nonnegative for $1 \le k \le 2$ and $1 \le j \le 3$.

Equations for planes can then be created using this notation. For example,

$$\begin{aligned} T_1(x, y) &= a_1 x + b_1 y + d_1 \\ &= (a_{11} - a_{12})x + (b_{11} - b_{12})y + (d_{11} - d_{12}) \end{aligned}$$

represents the equation of the first plane. For the sake of aesthetics, we prescribe that each plane has slopes in the x- and y-directions falling between -.1 and .1.

Costs, in dollars, associated with filling, cutting, and transporting are given as follows:

- The cost of removing one cubic foot of dirt from any rectangle is \$2
- The cost of adding one cubic foot of dirt from any rectangle is \$1
- The cost of transporting one cubic foot from one rectangle to another is \$1

1. Construct the objective function for this LP, which consists of total costs stemming from three sources: the digging of dirt from the rectangles, the adding of dirt to the rectangles, and the transporting of dirt between pairs of rectangles.
2. Now formulate a set of constraints for the LP. They arise from the following requirements:

   (a) There is a gap between the original height of each rectangle and the height of the new plane containing the rectangle. This gap equals the difference between the fill and cut amounts for the rectangle. A total of eight equations arises from this condition.

   (b) Each rectangle has an area, A_i. The net volume of dirt removed from that rectangle can be expressed in two ways. The first uses the quantities A_i, c_i, and f_i; the other utilizes the family of variables, $\{t_{ij}\}$. For example, the volume of dirt added to rectangle one, equals

   $$(t_{21} - t_{12}) + (t_{31} - t_{13}) + ... + (t_{81} - t_{18}),$$

   which can also be expressed in terms of A_1, c_1, and f_1. Eight more constraints arise from this condition.

(c) Six more constraints stem from the requirements that the slopes $a_j = a_{j1} - a_{j2}$ and $b_j = b_{j1} - b_{j2}$, where $1 \leq j \leq 3$, must fall within their prescribed lower and upper limits.

(d) The restriction that no gaps exist between planes is fulfilled if planes 1 and 3 intersect at both $(0, 40)$ and $(56, 40)$, planes 1 and 2 at both $(56, 100)$ and $(56, 40)$, and planes 2 and 3 at $(80, 40)$.

3. Now use Maple to solve the LP that minimizes the objective from (1), subject to the constraints from (2).

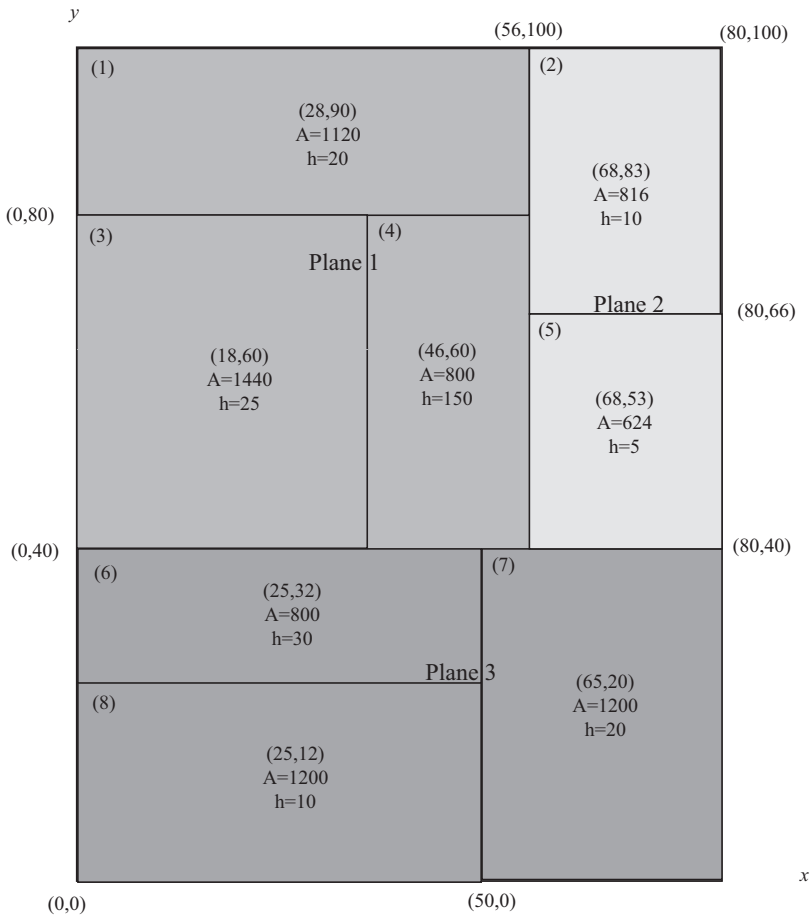


FIGURE A.1: Land tract site consisting of eight rectangles forming three planes.

A.2 The Juice Logistics Model

Transportation problems constitute one of the most common applications of linear programming. In this project, we investigate a modified version of one such problem faced by the Welch's grape processing company in the early 1990s.[2] At that time, Welch's sought to minimize costs associated with seasonal fluctuations in grape production, the transport of raw materials between plants, and storage costs from one growing season to the next. New policies, based upon linear programming, resulted in a net savings for Welch's of more than $130,000 during the first year of adoption.

Assume the company collects "raw," unprocessed grape juice, hereafter referred to simply as "juice," from grape growers at two different plants, labeled plant 1 ($k = 1$) and plant 2 ($k = 2$). The former is located in an urban setting, the latter in a rural region. The collection process takes place over a 12-month period, with month $i = 1$ corresponding to September, when the grape harvest is largest. The majority of grapes are collected from growers who belong to a regional Grape Cooperative Association (GCA). However, during the winter months, when local harvesting is low, the company frequently seeks grape sources elsewhere, including overseas.

Simultaneous to the collection of juice at the plants is the processing of two different products, jam ($j = 1$) and juice concentrate ($j = 2$). Company policy dictates that a minimum percentage of juice must be processed for each product during a given month at each particular plant. To ensure that enough juice is on hand, the company is free to transfer juice from one plant to another during any given month. Because of transportation costs, the company might be motivated to maintain a large juice inventory at each plant. The drawback of such a practice, however, is the carryover cost associated with the storage of unused juice into the next season.

The objective of the company is to minimize costs that stem from three different sources: cost associated with shipping finished products to stores, cost stemming from moving juice between the two plants, and cost due to storing juice at each plant.

We define the decision variables for this model using the following notation. Assume all amounts are measured in tons.

- $TS_{i,j,k}$, where $1 \leq i \leq 12$ and $1 \leq j, k \leq 2$: Amount of product j shipped to stores from plant k at month i.

[2] Based upon Schuster and Allen, [43], (1998).

- $TI_{i,k,m}$, where $1 \leq i \leq 12$ and $1 \leq k, m \leq 2$: Amount of grape juice transferred into plant k from plant m during month i. A stipulation, of course, is that $TI_{i,k,k} = 0$ for $1 \leq k \leq 2$ and $1 \leq i \leq 12$.

- $TO_{i,k,m}$, where $1 \leq i \leq 12$ and $1 \leq k, m \leq 2$: Amount of grape juice transferred out of plant k and into plant m during month i. Again, we assume that $TO_{i,k,k} = 0$ for $1 \leq k \leq 2$ and $1 \leq i \leq 12$. This family of decision variables may not appear necessary since we can interpret negative values of $TI_{i,k,m}$ as the amount of juice transferred out of plant k and into plant m, but adding these variables, along with appropriate constraints, permits us to construct a model in which all decision variables are nonnegative. For example, a negative value of $TI_{i,k,m} - TO_{i,k,m}$ means that plant k suffers a net loss of juice to plant m, even though each individual quantity in this difference is nonnegative.

- $EI_{i,k}$, where $1 \leq i \leq 12$ and $1 \leq k \leq 2$: The amount of grape juice inventory at plant k at end of month i.

Associated with the first three of these four families of variables are various costs, which are as follows:

- The cost of transporting juice from plant k to the other plant during month i is \$65 per ton.

- Manufacturing costs: For each plant, jam costs \$150 per ton to produce and juice concentrate \$175 per ton.

- The cost of storing juice in each plant from one season to the next: Assume this amount is \$500 per ton for either plant.

1. Formulate the objective function for this LP, which consists of the total cost of manufacturing the two products at the plants, transferring the juice from one plant to the other, and storing juice from one season to the next. Your function will involve triple summations.

   There are several important quantities that determine the LP's constraints.

   - At the start of every year, each plant is required to have 5 tons of juice on hand.
   - The amount of juice delivered to each plant at the start of month i is approximately

$$P_1(i) = 15\left(1 + \sin\left(\frac{\pi}{6}i\right)\right)$$

for plant 1, and

$$P_2(i) = 50\left(1 + \sin\left(\frac{\pi}{6}i\right)\right)$$

for plant 2. Note that the delivery amounts can be created as lists in Maple using the following syntax:

```
> P1:=[seq(15*(1+sin(Pi*i/6)),i=1..12)]:
> P2:=[seq(50*(1+sin(Pi*i/6)),i=1..12)]:
```

- At plant 1, 3 tons of juice are required to produce 1 ton of jam. At plant 2, 4 tons of juice are needed. At either plant, 2 tons of juice are required to create one ton of juice concentrate.
- Monthly product demands require that at least 10 tons of jam and 15 tons of juice concentrate be produced.
- At most 50 tons can be shipped from one plant to another during any given month.
- Each plant can produce at most 150 tons of the combined products each year.

2. Use the decision variables and model parameters to formulate each of the following families of constraints.
   (a) At the end of each year, the juice inventory at each plant must equal 70 tons.
   (b) During each month, the amount transferred into one plant equals the amount transferred out of the other.
   (c) Production of the products at the plants must meet monthly demand.
   (d) At most 50 tons can be shipped from one plant to the other during any given month.
   (e) Each plant is limited in how much jam and juice concentrate it can produce annually.
   (f) Each month, the amount of juice transferred into one plant equals the amount transferred out of the other.
   (g) A balance of juice exists from one month to the next.
      i. At the end of month 1, the ending inventory at each plant is the initial inventory of 70, plus the juice transferred in, less the juice transferred out, less juice lost to production of each product, plus juice obtained via normal delivery.

ii. At the end of month i, where $2 \leq i \leq 12$, the ending inventory is that from the previous month, plus the juice transferred in, less the juice transferred out, less juice lost to production of each product, plus juice obtained via normal delivery.

3. Now use Maple to solve the LP that minimizes the objective from (1), subject to the constraints from (2).

A.3 Work Scheduling with Overtime

Integer linear programming is frequently used to solve problems related to scheduling. In this project, we address such a problem, one faced by a state government seeking to meet prison guard staffing needs at a detention facility, in as cost-effective manner as possible. [3]

Each guard works five consecutive days per week and has the option of working overtime on one or both of his two days off. The larger the regular workforce, the more costly to the state are fringe benefit expenditures paid to the guards. On the other hand, the smaller the workforce, the more the state pays in overtime wages. The problem then becomes one of determining the workforce size and method of scheduling that meets staffing needs yet minimizes labor costs stemming from regular wages, overtime wages, and fringe benefits.

For the sake of simplicity, we focus on the 8 a.m. to 4 p.m. shift and for the sake of notational convention, we assume that the days of the week are numbered so that Monday corresponds to day 1, Tuesday to day 2, etc. Each prison guard is assigned to work the same 8-hour shift on five consecutive days, with the option of working the same shift on one or both of his following two days off. To say that a guard is assigned to work schedule k, means that his standard work week begins on day k.

We define the decision variables for this model using the following notation:

- x_k, where $1 \leq k \leq 7$: The number of guards assigned to work schedule k.
- u_k, where $1 \leq k \leq 7$: The number of guards assigned to work schedule k, who work overtime on their first day off.
- v_k, where $1 \leq k \leq 7$: The number of guards assigned to work schedule k, who work overtime on their second day off.

Staffing costs stem from wages, both regular and overtime, as well as fringe benefits and summarized as follows:

- The regular pay rate for guards is \$10 per hour, except on Sundays, when it is \$15 per hour instead.
- A guard who works on one or more of his or her two days off is paid at a rate of \$15 per hour on each of those days.

[3] Based upon Maynard, [31], (1980).

- Each guard receives fringe benefits amounting to 30% of his or her regular wages.

1. Construct the objective function for this ILP, which consists of total staffing costs due to regular wages, overtime wages, and fringe benefits.
2. Now formulate a set of constraints for the ILP. They arise from the following requirements:
   (a) Prison requirements stipulate at least 50 guards be present each day. The exception to this rule occurs on Sundays when 55 guards are required during family visitation.
   (b) To prevent guard fatigue on any given day, the number of prison guards working overtime can comprise no more than 25% of the total staff working that day.
   (c) To provide sufficient opportunities for guards to earn overtime wages, union rules dictate that on any given day, at least 10% of the total staff working that day consists of guards working overtime.
3. Now use Maple to solve the ILP that minimizes the objective from (1), subject to the constraints from (2).

A.4 Diagnosing Breast Cancer with a Linear Classifier

Breast cancer is the most common form of cancer and the second largest cause of cancer deaths among women. In the mid-1990s a noninvasive diagnostic tool was developed that can be described in terms of nonlinear programming.[4] Researchers use small-gauge needles to collect fluid from both malignant and benign tumors of a large number of women. With the aid of a computer program called **Xcyt**, the boundaries of the cell nuclei are analyzed and categorized with regard to a number of features, including area, perimeter, number of concavities, fractal dimension, variance of grey scale, etc. Mean values, variance, and other quantities for these attributes are computed, and ultimately 30 data items for each sample are represented by a vector in $\mathbb{R}^{30}$.

The set of all such vectors constitutes what is known as a "training set." Our aim is to construct a *hyperplane* in $\mathbb{R}^{30}$, using these training vectors, that "best separates" the training vectors corresponding to benign tumors from those corresponding to malignant ones. Figure A.2 illustrates this basic concept in $\mathbb{R}^2$, in which case the hyperplane consists of a line.

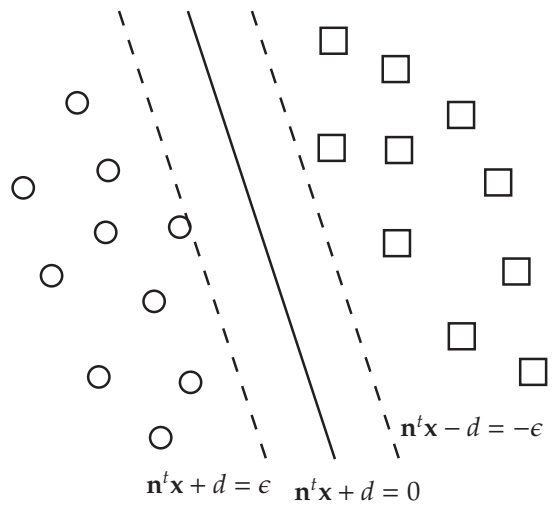


FIGURE A.2: Hyperplane consisting of a solid line that separates two classes of training vectors, circles and boxes, in $\mathbb{R}^2$.

This hyperplane can be used to classify, with a reasonable degree of certainty, whether a newly collected sample corresponds to a benign or malignant

[4]Based upon Mangasarian et al., [26], (1994).

tumor. Because the model consists of two classes and a hyperplane is used to separate the data, we refer to this prediction model as a *linear classifier*.

In this project we will construct the linear classifier by means of solving a constrained nonlinear programming problem. For the sake of simplicity and in order to visualize the underlying ideas better, we use 20 data vectors in $\mathbb{R}^3$ instead of $\mathbb{R}^{30}$. These vectors are expressed as ordered triples in Table A.1.

TABLE A.1: Training set

i	Training Vector, $\mathbf{x}_i$	y_i
1	(7.04, 6.52, 24.4)	1
2	(9.57, 4.42, 23.6)	1
3	(7.00, 8.30, 32.3)	1
4	(7.02, 6.57, 26.5)	1
5	(8.48, 3.62, 17.3)	1
6	(7.7, 5.98, 25.5)	1
7	(7.86, 5.57, 23.7)	1
8	(10.1, 5.97, 31.7)	1
9	(5.10, 5.82, 15.8)	1
10	(9.14, 5.60, 28.7)	1
11	(9.12, 7.33, 29.7)	−1
12	(7.29, 6.03, 18.9)	−1
13	(7.78, 7.65, 26.3)	−1
14	(7.92, 6.67, 27)	−1
15	(7.69, 5.37, 19.6)	−1
16	(8.54, 5.33, 21.8)	−1
17	(9.43, 4.85, 21)	−1
18	(6.97, 5.09, 15.9)	−1
19	(6.97, 5.7, 17)	−1
20	(9.26, 2.84, 14.4)	−1

To the training vector, $\mathbf{x}_i$, where $1 \leq i \leq 20$, we associate a value $y_i \in \{-1, 1\}$, where $y_i = 1$ (resp. $y_i = -1$) if that training vector corresponds to a benign (resp. malignant) tumor.

In $\mathbb{R}^3$ the hyperplane takes the form of a plane, $n_1x_1 + n_2x_2 + n_3x_3 + d = 0$, for some yet-to-be-determined real quantities, n_1, n_2, n_3, and d. If we write $\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}$ (the normal vector) and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, this equation is identical to $\mathbf{n}^t\mathbf{x} + d = 0$.

Ideally all malignant tumor vectors fall on one side of this plane and all benign vectors fall on the other. In this case, there exists some positive constant ϵ such that $\mathbf{n}^t\mathbf{x}_i + d \geq \epsilon$ for $1 \leq i \leq 10$ and $\mathbf{n}^t\mathbf{x}_i + d \leq -\epsilon$ for $11 \leq i \leq 20$. We can think of ϵ as the smallest possible distance between any of the training vectors and the desired plane. It determines two new planes, $\mathbf{n}^t\mathbf{x} + d = \epsilon$ and $\mathbf{n}^t\mathbf{x} + d = -\epsilon$,

as depicted in Figure A.2. The plane, $\mathbf{n}^t\mathbf{x} + d = 0$, separates these two new planes and is a distance ϵ from each.

Since $y_i = 1$ for $1 \le i \le 10$ and $y_i = -1$ for $11 \le i \le 20$, the preceding two families of inequalities may be combined into one and written as

$$y_i\left(\mathbf{n}^t\mathbf{x}_i + d\right) \ge \epsilon \quad \text{for } 1 \le i \le 20. \tag{A.1}$$

Of course, by scaling $\mathbf{n}$ and d by $\frac{1}{\epsilon}$ if necessary, we may rewrite (A.1) as follows:

$$y_i\left(\mathbf{n}^t\mathbf{x}_i + d\right) \ge 1 \quad \text{where } 1 \le i \le 20. \tag{A.2}$$

Thus, the three planes in Figure A.2 can be expressed as

$$\mathbf{n}^t\mathbf{x} + d = -1,\ \mathbf{n}^t\mathbf{x} + d = 0, \text{ and } \mathbf{n}^t\mathbf{x} + d = 1.$$

1. The *margin* is defined to be distance between the two planes, $\mathbf{n}^t\mathbf{x} + d = 1$ and $\mathbf{n}^t\mathbf{x} + d = -1$. Because d merely denotes a translation, the margin is the same as the distance between $\mathbf{n}^t\mathbf{x} = 1$ and $\mathbf{n}^t\mathbf{x} = -1$. It is this quantity that we see to maximize, or equivalently, its reciprocal that we seek to minimize. Show that the margin equals $\frac{2}{\|\mathbf{n}\|}$. (Hint: Choose arbitrary vectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, that satisfy $\mathbf{n}^t\mathbf{x}_1 = 1$ and $\mathbf{n}^t\mathbf{x}_2 = -1$. Then determine the orthogonal projection of $\mathbf{x}_2 - \mathbf{x}_1$ onto $\mathbf{n}$. The magnitude of this projection is the distance between the two planes.)

   It is very unlikely that we can determine a plane that completely "separates" the benign training vectors from the malignant ones, in which case the constraints

   $$y_i\left(\mathbf{n}^t\mathbf{x}_i + d\right) \ge 1 \quad \text{for } 1 \le i \le 20. \tag{A.3}$$

   lead to an infeasible NLP. To address this problem, we relax each constraint and rewrite (A.3) as

   $$y_i\left(\mathbf{n}^t\mathbf{x}_i + d\right) \ge 1 - \delta_i \quad \text{for } 1 \le i \le 20, \tag{A.4}$$

   where $\delta_i \ge 0$ denotes the amount of violation in constraint i. The family, $\{\delta_i \mid 1 \le i \le 20\}$, together with d and the components of $\mathbf{n}$, then leads to a total of 24 decision variables for this model, 20 of which we require to be nonnegative.

   Of course, we must ensure the total penalty violation is as small as possible.

2. Construct a penalty function that is large if the total constraint violation is large.

3. Use Maple to solve the NLP that arises by minimizing the sum of $\frac{\|\mathbf{n}\|}{2}$ and the penalty function, subject to the constraints in (A.4) together with the sign restrictions $\delta_i \geq 0$ for $i = 1, 2, \ldots, 20$.

4. The solution to your NLP determines the equation of the desired plane. Use Maple to plot this plane, together with the training vectors. To do this, create two `pointplot3d` structures, one for each of the two vector types. (Specifying different symbols, e.g., `symbol=solidcircle` and `symbol=solidbox` for the two plots will highlight the difference between the two training vector types.) Then create a third plot structure, this one producing the separating plane. Combine all three plot structures with the `display` command.

5. Suppose two new sample vectors are given by $\mathbf{v}_1 = \begin{bmatrix} 10.1 \\ 5.97 \\ 31.7 \end{bmatrix}$ and $\mathbf{v}_2 = \begin{bmatrix} 8.54 \\ 5.33 \\ 21.8 \end{bmatrix}$. Use your newly constructed linear classifier, $f : \mathbb{R}^3 \to \mathbb{R}$ given by $f(\mathbf{x}) = \mathbf{n}^t\mathbf{x} + d$, to classify each of $\mathbf{v}_1$ and $\mathbf{v}_2$ as corresponding to benign or malignant vector type.

A.5 The Markowitz Portfolio Model

The Markowitz Portfolio Model, created by economist Harry Max Markowitz, provides a means of maximizing return on an investment portfolio while at the same time accounting for an investor's attitude toward risk.[5]

Suppose we wish to construct a portfolio consisting of four stocks and a certificate of deposit.

1. Pick four stocks and determine the prices of each over an extended period of time. For example, you may wish to determine the monthly closing price of the stock over a period of several years. Various means exist for acquiring such data. Internet resources, such as Yahoo! Finance, allows the user to specify the dates of interest for a given stock and whether the closing prices are to be sampled daily, weekly, or monthly. Data can then be downloaded in Excel format.

   Maple's `ExcelTools` package provides the means to import this data into a Maple worksheet. Suppose closing prices for the first stock have been saved in cells A1 to A100 of a sheet labeled "stock1" within the Excel file, "stockdata.xls" and that this file is located in the "C:\\Users" directory. Then the following syntax imports this data into an 100 by 1 array, `S`:

   ```
   > with(ExcelTools):
   > S:=Import("C:\\Users\\","stockdata.xls","stock1", "A1:A100"):
   ```

   For each stock, we wish to compute its average rate of return per unit time period. For example, if the 100 values described above are collected on the last day of each month, then the rate of return for month j is given by

   $$r_j = \frac{\text{closing price for month } j\text{-closing price for month } (j-1)}{\text{closing price for month } (j-1)}. \quad \text{(A.5)}$$

2. For each stock you selected, compute the corresponding rates of return.

   The mean and variance of a list of numbers, $\{r_1, r_2, \ldots, r_N\}$, are given by

   $$\mu = \frac{1}{N}\sum_{j=1}^{N} r_j \quad \text{and} \quad \sigma^2 = \frac{1}{N}\sum_{j=1}^{N}(r_j - \mu)^2,$$

[5] Based upon Straffin, [45], (1996).

respectively. The second of these measures the extent to which the data deviates from the mean. Thus, a larger variance corresponds to more "volatility" in the data.

3. For each of the stocks, compute the corresponding mean and variance of the rates of return. One way to do this in Maple is to create a list of these rates of return, one list for each stock. The `Mean` command takes each list as its argument and computes the corresponding mean. The `Variance` command works similarly. Both commands are located in the `Statistics` package. (Note: At this stage, it is highly desirable to make sure at least one of the stocks has a positive mean rate of return. If this is not the case, then we should replace one of our stocks with one having this property.)

4. Suppose that, in addition to the four stocks, we wish to include a risk-free investment in our portfolio, such as a certificate of deposit (CD). If we use a CD, then we must determine its rate of return per unit time period, where the time period is the same as that used for the stocks, be it daily, monthly, etc. Select a CD, whose *annual percentage yield* (APY) is known to you. Use the compound interest formula to determine the rate of return per unit time period that corresponds to this APY. Call the resulting value, μ_5. Then show that the corresponding variance is zero, which makes sense for a risk-free investment.

With means and variances in hand for the five different investments, we are now in a position to construct a portfolio. Suppose we let x_i, where $1 \leq i \leq 5$, denote the fraction of our available funds we will devote to investment i. Given a sense of how much risk we are willing to tolerate, we seek to determine values of these weights that maximize the return on our portfolio.

Suppose we had used specific values of these weights to construct a portfolio for the time period corresponding to the original data. Then our rate of return of the portfolio for that given time period would have been

$$R_j = x_1 r_{1,j} + x_2 r_{2,j} + x_3 r_{3,j} + x_4 r_{4,j} + x_5 r_{5,j}.$$

where $r_{i,j}$ denotes the rate of return during time period j for investment type i. In other words, the daily rate of return for the portfolio would have been merely the weighted sum of the individual rates of return of the five investment types. Suppose we assume that the stocks are diverse enough in nature the rates of return for any pair of them are uncorrelated. Then, elementary statistics dictates that for such a weighted sum, the portfolio mean and variance of $\{R_1, R_2, \ldots, R_N\}$ are given by

$$\mu_p = x_1\mu_1 + x_2\mu_2 + x_3\mu_3 + x_4\mu_5 + x_5\mu_5$$

and

$$\sigma_p^2 = x_1^2\sigma_1^2 + x_2^2\sigma_2^2 + x_3^2\sigma_3^2 + x_4^2\sigma_4^2 + x_5^2\sigma_5^2. \tag{A.6}$$

While past performance of a stock is no guarantee of how it will fare in the future, we can still use the mean and variance in (A.6) to estimate the performance and volatility of a portfolio for a given set of weights.

The goal of the investor is to maximize μ_p and, simultaneously, minimize σ_p^2. As these are competing goals, we create a single objective function, one that seeks to determine weights, x_1, x_2, x_3, x_4, x_5, that maximize μ_p while assigning a penalty to large values of σ_p^2. The magnitude of the penalty is a function of the investor's risk tolerance. We use the term *risk aversion parameter* to describe this quantity and denote it by α.

We therefore seek to solve the following NLP:

$$\begin{aligned} \text{maximize } f(x_1, x_2, x_3, x_4, x_5) &= \sum_{i=1}^{5} x_i\mu_i - \alpha \sum_{i=1}^{5} x_i^2\sigma_i^2 \qquad \text{(A.7)} \\ \text{subject to } \sum_{i=1}^{5} x_i &= 1 \\ \text{and } x_i \geq 0 \text{ for } 1 \leq i \leq 5. & \end{aligned}$$

5. The nonlinear programming model (A.7) is a quadratic programming problem. By defining a new function, f, express it in the standard form:

$$\begin{aligned} \text{minimize } f(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^t Q\mathbf{x} + \mathbf{p}^t\mathbf{x} \qquad \text{(A.8)} \\ \text{subject to} & \\ A\mathbf{x} &= \mathbf{b} \\ C\mathbf{x} &\leq \mathbf{d}, \end{aligned}$$

where $\mathbf{x}$ belongs to $\mathbb{R}^5$ and the remaining matrices and vectors are of appropriate dimensions. The matrix, Q, is a diagonal matrix depending upon α and the individual investment variances.

6. Experiment with various risk aversion parameters and solve (A.8) for each value you choose. Describe what happens to the weights and objective function value as α increases, and provide an economic interpretation of the nonzero Lagrange multipliers.

A.6 A Game Theory Model of a Predator-Prey Habitat

In the desert, snakes prey on small rodents in both open and vegetated habitats.[6] Should the rodent and snake elect to reside in the same habitat, we expects the two species to have negative and positive payoffs, respectively, due to potential inter-species encounters. Similarly, when the rodent and snake reside in different habitats, the payoffs are positive and negative, respectively.

Let habitats 1 and 2 correspond to the vegetated and open areas. (Hereafter, we simply refer to the vegetated area as the "bush.") Corresponding to the snake, we define the 2-by-2 payoff matrix A, where $[A]_{ij}$ represents the snake's payoff when it resides in habitat i and the rodent in habitat j. For the rodent, we denote its payoff matrix B, where $[B]_{ij}$ denotes the rodent's payoff when it resides in habitat i and the snake in habitat j. We expect that the diagonal entries of A to be nonnegative and those of B nonpositive.

Payoff for each species is defined in terms of dimensionless "energy gains" or "energy losses," whose relative sizes reflect the extent to which they affect a specie's future reproductive output. Entries for the two payoff matrices depend upon various parameters, such as probabilities of inter-species encounters in the two habitats. Table A.2 lists the various parameter and probability values as estimated by data collected in the field.

TABLE A.2: Field data used to determine payoff matrices

Quantity	Label	Value
Energy loss for rodent if captured by snake	d	1500
Energy gain for snake if it captures rodent	e	1000
Energy required of rodent to live in either habitat	α	7
Energy required of snake to live in either habitat	β	7
Probability snake captures rodent in the bush	P_{sb}	.032
Probability snake captures rodent in the open	P_{so}	.009
Energy gain for rodent that survives predation by snake in the bush	ρ_b	12
Energy gain for rodent that survives predation by snake in the open	ρ_o	17

1. Use the values of e, β, P_{sb}, and P_{so} to construct the snake's payoff matrix, A.
2. Use d, α, P_{sb}, P_{so}, ρ_b, and ρ_o to construct the rodent's payoff matrix, B.

[6] Based upon Bouskila, [9], (2001).

3. Determine the mixed strategy Nash equilibrium for this bimatrix game model of the predator-prey habitat.

4. Suppose the snake's average payoff must increase by 10% from its equilibrium value and that the rodent's average loss must decrease by 10%. What fraction of time should each species spend in each habitat to bring this change about?

Appendix B

Important Results from Linear Algebra

What follows is a list of results from linear algebra that are referenced at various places in this text. Further elaboration on each of them can be found in a variety of other sources, such as [15] and [21]. We assume in this list of results that all matrices have real-valued entries.

B.1 Linear Independence

Suppose that $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ is a set of vectors in $\mathbb{R}^m$ and that A is the m-by-n matrix having these vectors as columns. Then V is a set of linear independent vectors if and only if the homogeneous matrix equation $A\mathbf{x} = \mathbf{0}$ has only the trivial solution, $\mathbf{x} = \mathbf{0}$.

B.2 The Invertible Matrix Theorem

Theorem B.2.1. Suppose that A is an n-by-n matrix. Then the following conditions are logically equivalent.

1. The determinant of A is nonzero.
2. The reduced row echelon form of A is I_n, the n-by-n identity matrix.
3. A is invertible.
4. The set of column vectors of A and the set of row vectors of A both span $\mathbb{R}^n$.
5. The set of column vectors of A and the set of row vectors of A both form linear independent sets.
6. The set of column vectors of A and the set of row vectors of A both form a basis for $\mathbb{R}^n$.

7. The homogeneous matrix equation $A\mathbf{x} = \mathbf{0}$ has only the trivial solution, $\mathbf{x} = \mathbf{0}$.
8. For every vector $\mathbf{b}$ in $\mathbb{R}^n$ the matrix equation $A\mathbf{x} = \mathbf{b}$ has a unique solution given by $\mathbf{x} = A^{-1}\mathbf{b}$.
9. The column space of A has dimension n and the null space of A has dimension zero.
10. Zero is not an eigenvalue of A.

B.3 Transpose Properties

The transpose of an m-by-n matrix A, written A^t, is the n-by-m matrix obtained by interchanging the rows and columns of A. Important properties of the transpose include the following:

1. $(A^t)^t = A$.
2. $(A + B)^t = A^t + B^t$, for any other m by n matrix, B.
3. $(AB)^t = B^t A^t$, provided the matrix product AB is defined.
4. $\det(A^t) = \det(A)$.
5. If A is a square, invertible matrix, then so is A^t, and $\left(A^t\right)^{-1} = \left(A^{-1}\right)^t$.
6. If $\mathbf{u}$ and $\mathbf{v}$ are column vectors in $\mathbb{R}^n$, then $\mathbf{u}^t\mathbf{v} = \mathbf{v}^t\mathbf{u}$. In particular, if $\mathbf{u}$ is a column vector in $\mathbb{R}^n$, then $\mathbf{u}^t\mathbf{u} = \|\mathbf{u}\|^2$.
7. The Cauchy-Schwartz Inequality: $\left|\mathbf{u}^t\mathbf{v}\right| \leq \|\mathbf{u}\|\|\mathbf{v}\|$ for every pair of column vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^n$, with equality occurring if and only if $\mathbf{u}$ and $\mathbf{v}$ are scalar multiples of one another.
8. If A is an n-by-n matrix, then the eigenvalues of A^tA are all nonnegative. For a symmetric matrix, A, $A^tA = A^2$. Since each eigenvalue of A^2 is the square of an eigenvalue of A, it follows that the eigenvalues of a symmetric matrix are all real-valued.

B.4 Positive Definite Matrices

A symmetric, n-by-n matrix, Q, is positive definite if and only if $\mathbf{x}^tQ\mathbf{x} > \mathbf{0}$ for all $\mathbf{x}$ in $\mathbb{R}^n$. Equivalently, Q is positive definite if and only if its eigenvalues are

all positive. The matrix Q is positive semidefinite provided $\mathbf{x}^t Q\mathbf{x} \geq \mathbf{0}$ for all $\mathbf{x}$ in $\mathbb{R}^n$ or if all eigenvalues of Q are nonnegative. The terms negative definite and negative semidefinite are defined analogously.

B.5 Cramer's Rule

Suppose that A is a square, n-by-n matrix having a nonzero determinant. If $\mathbf{b}$ belongs to $\mathbb{R}^n$ and if $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, then the solution to the system of equations $A\mathbf{x} = \mathbf{b}$ can be expressed as

$$x_i = \frac{\det(A_i(\mathbf{b}))}{\det(A)},$$

where $1 \leq i \leq n$ and where $A_i(\mathbf{b})$ denotes the matrix obtained by replacing column i of A with the vector $\mathbf{b}$.

B.6 The Rank-Nullity Theorem

Theorem B.6.1. If A is an m-by-n matrix, then the dimension of the column space of A added to the dimension of the null space of A equals n.

B.7 The Spectral Theorem

Theorem B.7.1. Suppose A is a symmetric matrix having real-valued entries. Then there exists an n-by-n diagonal matrix, D, and an n-by-n matrix, P, having the property that $A = PDP^t$. The matrix D can be chosen to have the eigenvalues of A along its diagonal. The matrix P is then formed using as its columns the corresponding eigenvectors of A, normalized so as to have length one. Finally, P is invertible, with $P^{-1} = P^t$.

B.8 Matrix Norms

Let $M_n(\mathbb{R})$ denote the set of n-by-n matrices having real-valued entries. Then $M_n(\mathbb{R})$ forms a vector space under the usual operations of matrix addition and scalar multiplication. A norm on $M_n(\mathbb{R})$ is a function that assigns to each matrix, A, in this set, a nonnegative real number, $||A||$. This function must satisfy the usual properties of a vector space norm:

1. For each A in $M_n(\mathbb{R})$, $||A|| \geq 0$, with equality possible only if A is the zero matrix.
2. For each A in $M_n(\mathbb{R})$ and each real scalar, c, $||cA|| = |c|||A||$.
3. For every A and B in $M_n(\mathbb{R})$, $||A + B|| \leq ||A|| + ||B||$.
4. For every A and B in $M_n(\mathbb{R})$, $||AB|| \leq ||A||||B||$.

The third of these properties is referred to as the triangle inequality; the last is referred to as the sub-multiplicative property and is analogous to the Cauchy-Schwartz Inequality for vectors.

While several different matrix norms exist for $M_n(\mathbb{R})$, the one connects the Euclidean norm of a vector, $\mathbf{x}$, in $\mathbb{R}^n$ with that of the matrix-product vector, $A\mathbf{x}$ is known as the *spectral norm*. It is defined as

$$||A|| = \max\left\{ \sqrt{\lambda} \mid \lambda \text{ is an eigenvalue of } A^tA \right\}.$$

Note that by the last property from B.3, each eigenvalue in this set is non-negative. The spectral norm of a matrix A in $M_n(\mathbb{R})$ satisfies the vector-norm inequality

$$||A\mathbf{x}|| \leq ||A||||\mathbf{x}|| \text{ for all } \mathbf{x} \in \mathbb{R}^n.$$

In other words, multiplication of $\mathbf{x}$ by A yields a vector whose length is no more than that of $\mathbf{x}$, scaled by the matrix norm, $||A||$.

When A is symmetric, the eigenvalues of AA^t are the eigenvalues of A^2, and the spectral norm reduces to $||A|| = \rho(A)$, where $\rho(A)$ is the *spectral radius*, or maximum of the absolute values of the eigenvalues of A. Hence, for a symmetric matrix, A,

$$||A\mathbf{x}|| \leq \rho(A)||\mathbf{x}|| \text{ for all } \mathbf{x} \in \mathbb{R}^n.$$

Appendix C

Getting Started with Maple

C.1 The Worksheet Structure

Maple is an interactive, mathematical problem solving software package well-suited for addressing problems that arise in this text. Perhaps this software's best feature is its versatility. Capabilities of Maple fall into four broad categories.

1. Maple has a symbolic computing component. It can manipulate algebraic expressions, solve equations exactly, perform calculus and linear algebra computations, and so on. The list of symbolic computing capabilities is quite extensive.

2. The software has built-in commands for performing numeric computations, such as those leading to solutions of differential equations.

3. Maple possesses the ability to produce high-quality graphical output, including those involving functions of one or two variables, implicitly defined functions, and vector quantities.

4. Maple is a programming language in that it is capable of combining all its tools in a programming environment that is intuitive in nature and that requires little to no programming experience.

Maple performs related tasks in what is referred to as a *Maple worksheet*. Each worksheet is comprised of command lines starting with > and can be saved for future using the `.mw` file name extension. While the input and output of each worksheet are visible when a previously created file is opened, values in a worksheet are not saved. Two options exist for re-executing a worksheet:

1. Execute, in the order presented, each command line of the worksheet by placing the cursor anywhere in that command line and hitting "Enter."

2. Execute the entire worksheet at once, either by selecting the Edit-Execute command or by selecting the !!! button at the top of the worksheet.

Maple follows certain conventions and requires the user to observe various rules. Among those that are most important are the following:

1. Maple is case-sensitive.
2. Every Maple command line ends with either a semicolon (;) or colon (:), and the command line is executed by hitting the "Enter" key.
3. When a semicolon is used, output from the command is visible to the user. When a colon is used instead, the command is executed but no output appears. Using a colon is especially helpful when one wishes to perform lengthy calculations, whose results are to be used later in a worksheet but are not important for immediate viewing.
4. Variables and other types of structures are assigned values in Maple using the colon-equal sign combination. For example, to assign the value of 2 to the variable a, type `a:=2;` at the command line.
5. Maple "remembers," in chronological order, how commands within a worksheet are executed. For example, if the variable `a` is assigned the value of 3 at some command line in the worksheet, and the user executes a subsequent command line that involves `a`, this subsequent command assumes that $a = 3$. Two means exist to "clear" variable values. To clear all values in a worksheet, type `restart` at the first command line and re-execute the worksheet. To clear a specific variable value, such as a, type `a:='a';` at the command line.
6. Frequently one wishes to insert additional lines at some point in a worksheet. If the line to be added is below the current cursor position, type "CTRL-J." If the line to be added is above, type "CTRL-K" instead.
7. Maple permits copying and pasting. Short-cut keys for copy and paste are "CTRL-C" and "CTRL-V," respectively.
8. Maple provides the means for the user to enter ordinary text using its "text mode." To switch to this mode, select the "T" button at the top of the worksheet or type "CTRL-T."
9. To add documentation to any line of a worksheet, insert a line of text as described previously or type, anywhere in the command line, the "#" followed by the desired comment. Maple will then ignore all input that follows # in that line.
10. To learn appropriate syntax and to view examples using a particular Maple command, type `?command name` at the command prompt.

With this general background information in mind, we now focus on using Maple to perform specific tasks.

C.2 Arithmetic Calculations and Built-In Operations

Arithmetic in Maple can be performed on either fixed numeric or variable quantities. Multiplication requires the use of * and exponentiation the use of the ^ symbol. Some examples include the following:

```
> a:=8;
```

$$a := 8$$

```
> 3*a;
```

$$24$$

```
> 2^5;
```

$$32$$

```
> 4/a;
```

$$\frac{1}{2}$$

Commonly used built-in functions are those used to compute square roots, logarithms, exponentials, and trigonometric values. Note how the exponential e^x is written, `exp(x)`, and the constant π is written as `Pi`.

```
> x:=9:
> sqrt(x);
```

$$3$$

```
> ln(1);
```

$$0$$

```
> exp(2);
```

$$e^2$$

```
> sin(Pi/3);
```

$$\frac{1}{2}\sqrt{3}$$

Maple returns exact values where possible. To obtain a floating-point representation of a number, use the `evalf` command. For example, `evalf(Pi)` returns 3.141592654. The default number of digits Maple uses for floating point values is 10. To change this number, at the start of a worksheet enter the command `Digits:=N;`, where `N` is the number of desired digits.

Frequently one wishes to take output from one command line and use it directly in the next line without retyping the output value or assigning it a name. The `%` symbol is useful for such situations in that it uses the most recent output as its value.

```
> theta:=Pi/3;
```

$$\theta := \frac{1}{3}\pi$$

```
> sin(theta);
```

$$\frac{1}{2}\sqrt{3}$$

```
> evalf(%);
```

0.8660254040

A word of warning is in order regarding the use of %: Its value is the output from the most recently executed command line. This value may or may not be the output from the previous line in the worksheet, depending upon whether or not command lines are executed in the order in which they appear. For this reason, use % with caution.

C.3 Expressions and Functions

Expressions involving variables can be assigned names and manipulated by passing the assigned name as an argument to the appropriate command. Typical operations performed on expressions include factoring and simplifying, as well as substitution of values (via the subs command) and equation solving. For systems of equations, the equations themselves are listed within braces, as are the unknown variables. Here are more examples:

```
> y:=(x+1)*(x-3);
```

$$y := (x+1)(x-3)$$

```
> expand(y);
```

$$x^2 - 2x - 3$$

```
> z:=t^2-t-6;
```

$$z := t^2 - t - 6$$

```
> factor(z);
```

$$(t-3)(t+2)$$

```
> subs(t=1,z);
```

$$-6$$

```
> y:=x^2+3*x+1;
```

$$y := x^2 + 3x + 1$$

```
> solve(y=0,x);
```

$$-\frac{3}{2} + \frac{1}{2}\sqrt{5}, \quad -\frac{3}{2} - \frac{1}{2}\sqrt{5}$$

```
> fsolve(y=0,x);
```

$$-2.618033989, -.3819660113$$

```
> solve({y=2x+3,y=-x+4},{x,y});
```

$$\left\{x = \frac{1}{3}, y = \frac{11}{3}\right\}$$

As a general rule, `solve` returns exact roots, both real- and complex-valued of an equation or system of equations, to the extent Maple is capable of doing so. If unable to compute all roots, Maple will return a warning of the form `warning: SolutionsMayBeLost`. Sometimes, Maple will express certain solutions using placeholders of the form `RootOf`. This is especially true if one root is difficult for Maple to compute, yet a second root is dependent upon the first. Here is an example:

```
> solve({y=2*x, x^4-x^2},{x,y});
```

$$\left\{x = \text{RootOf}(Z^4 - Z^3 + 1, \text{label} = \text{L1}), y = 2\text{RootOf}(Z^4 - Z^3 + 1, \text{label} = \text{L1})\right\}$$

This output indicates that each solution value x, of which there are four, corresponds to a value of y, each of which is twice x.

Maple's `fsolve` command differs from `solve` in that approximate solutions are returned. For single equations having real-valued coefficients, `fsolve` returns approximations of all real-valued roots. For systems of such equations, Maple returns one approximate solution. Further details regarding the `solve` and `fsolve` commands can be found by typing `?solve/details` or `?fsolve/details` at the command prompt.

Functions can be entered in Maple using arrow notation (a dash followed by a greater-than sign) or via the `unapply` command. For example, to enter the function $f(x) = x^2 + 1$, type either `f:=x->x^2-4` or `f:=unapply(x^2-4,x)`. Once a function is defined, it can be used to evaluate both numeric and variable outputs.

```
> f:=x->x^2-4;
```

$$f := x \to x^2 - 4$$

```
> f(3);
```

$$5$$

```
> f(3+h);
```

$$(3+h)^2 - 4$$

```
> expand(%);
```

$$5 + 6h + h^2$$

Functions of more than one variable are entered in a very similar manner.

```
> f:=(x1,x2)->sin(x1+x2);
```

$$f := (x1, x2) \to \sin(x_1 + x_2)$$

```
> f(x1,Pi);
```

$$\sin(x_1 + \pi)$$

A very important distinction exists in Maple between functions and expressions. The rule `f` defined using the `->` notation or `unapply` command is a function, whereas quantities such as `f(x)` and `f(3+h)` are expressions in x and h, respectively. Commands for expressions may be applied to `f(x)` and `f(3+h)`. For example, `solve(f(x)=0,x)` determines the roots of the function f and `factor(f(x))` produces the factored form of $f(x)$, which is again an expression.

Functions are more appropriate than expressions when one wishes to perform "function-like" operations, such as evaluating function values or creating new functions from old ones. For example, to compute the derivative function, use the Maple `D` operator. Here is an example utilizing this operator to calculate first and second derivatives of a function, along with critical points and inflection points and their respective function outputs. Note the documentation through the use of the `#`.

```
> f:=x->x*exp(x);
```

$$f := x \to xe^x$$

```
> D(f)(x); # Calculate derivative function.
```

$$e^x + xe^x$$

```
> evalf(D(f)(1)); # Calculate derivative value at x=1.
```

$$5.436563656$$

```
> solve(D(f)(x)=0,x); # Determine critical point of function.
```

$$-1$$

```
> f(%); # Evaluate function at critical point.
```

$$-e^{-1}$$

```
> (D@@2)(f)(x); # Calculate second derivative function.
```

$$2e^x + xe^x$$

```
> solve((D@@2)(f)(x)=0,x); # Determine point of inflection of function.
```

$$-2$$

```
> f(%);# Evaluate function at inflection point.
```

$$-2e^{-2}$$

The commands `diff(f(x),x)` and `diff(f(x),x$2)` also yield the first and second derivatives as expressions and could be combined with the `solve` command to determine the preceding critical point and inflection point.

C.4 Arrays, Lists, Sequences, and Sums

An array in Maple can be considered as a table of dimension $n \geq 1$, in which each entry corresponds to an ordered n-tuple. Arrays are created using the `array` command, and entries are assigned values through a nested `for-do` loop structure. This simple programming structure takes the form

```
> for i from m to n do task;od;
```

where *m* and *n* represent the smallest and largest index values, respectively, and *task* denotes the operation to be performed for index value, `i`.

For example, here is a 2-by-3 array, in which each entry is assigned the sum of the corresponding column and row indices. The entire array is then printed.

```
> A:=array(1..2,1..3);
```

$$A := array(1..2, 1..3, [])$$

```
> for i from 1 to 2 do for j from 1 to 3 do A[i,j]:=i+j:od:od:
```

```
> print(A);
```

$$\begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

Later we will see how arrays provide a convenient means for creating variables labeled with double or triple indices.

Lists are merely one-dimensional arrays. The only advantage of a list is that it need not be first defined using the `array` command. For example, the command `L:=[1,2,4,8,16]` defines a list, such that `L[1]=1`, `L[2]=2`, and so on. A sequence, created using the `seq` command, can be thought of as a list without brackets. It is frequently used to create longer sequences of values as in the following commands, which create a sequence, called `S`, consisting of the first 10 nonnegative powers of two. This sequence is then used to form a list, labeled `T`.

```
> S:=seq(2^j,j=0..10);
```

$$S := 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024$$

```
> T:=[S];
```

$$T := [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]$$

To extract a sequence from a list, `T`, we use the command `op(T)`, and to determine the number of entries of this list, we use `nops(T)`. Here is an example:

```
> T:=[1,2,4,8,16,32,64];
```

$$T := [1, 2, 4, 8, 16, 32, 64]$$

```
> op(T);
```

$$1, 2, 4, 8, 16, 32, 64$$

```
> nops(T);
```

$$7$$

The Maple commands `sum` and `add` are used to add finite sequences of values together. The first command is used to determine the closed form representation, if one exists, of a sum involving symbolic quantities. The `add` command is used instead to add explicitly a finite sequence of values. The following examples illustrate the commands and their differences:

```
> sum(r^k,k=0..n);
```

$$\frac{r^{n+1}}{r-1} - \frac{1}{r-1}$$

```
> add(2^k,k=0..5);
```

$$63$$

C.5 Matrix Algebra and the `LinearAlgebra` Package

"Packages" in Maple contain commands that are made available to the user only if the package is loaded into the worksheet. This is accomplished through typing `with(`*`package name`*`)` at the command prompt. To load a package without seeing all package commands, use a colon at the end of the `with` command line; to see all commands listed, use a semicolon. Maple packages used in this text include `LinearAlgebra`, `plots`, `plottools`, `Optimization`, and `VectorCalculus`. To learn about a package's particular command, its uses, and appropriate syntax, type `?`*`command name`*.

Maple's `LinearAlgebra` package is capable of performing a wide variety of operations on matrices containing both numeric and symbolic entries. Here the package is loaded in a manner that reveals all its commands:

```
> restart;
> with(LinearAlgebra);
```

[*Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, *BilinearForm*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*, *ColumnDimension*, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*, *ConditionNumber*, *ConstantMatrix*, *ConstantVector*, *Copy*, *CreatePermutation*, *CrossProduct*, *DeleteColumn*, *DeleteRow*, *Determinant*, *Diagonal*, *DiagonalMatrix*, *Dimension*, *Dimensions*, *DotProduct*, *EigenConditionNumbers*, *Eigenvalues*, *Eigenvectors*, *Equal*, *ForwardSubstitute*, *FrobeniusForm*, *GaussianElimination*, *GenerateMatrix*, *Generic*, *GetResultDataType*, *GetResultShape*, *GivensRotationMatrix*, *HankelMatrix*, *HermiteForm*, *HermitianTranspose*, *HessenbergForm*, *HilbertMatrix*, *HouseholderMatrix*, *IdentityMatrix*, *IntersectionBasis*, *IsDefinite*, *IsOrthogonal*, *IsSimilar*, *IsUnitary*, *JordanBlockMatrix*, *JordanForm*, *LA_Main*, *LUDecomposition*, *LeastSquares*, *LinearSolve*, *Map*, *Map2*, *MatrixAdd*, *MatrixExponential*, *MatrixFunction*, *MatrixInverse*, *MatrixMatrixMultiply*, *MatrixNorm*, *MatrixPower*, *MatrixScalarMultiply*, *MatrixVectorMultiply*, *MinimalPolynomial*, *Minor*, *Modular*, *Multiply*, *NoUserValue*, *Norm*, *Normalize*, *NullSpace*, *OuterProductMatrix*, *Permanent*, *Pivot*, *PopovForm*, *QRDecomposition*, *RandomMatrix*, *RandomVector*, *Rank*, *RationalCanonicalForm*, *ReducedRowEchelonForm*, *Row*, *RowDimension*, *RowOperation*, *RowSpace*, *ScalarMatrix*, *ScalarMultiply*, *ScalarVector*, *SchurForm*, *SingularValues*, *SmithForm*, *StronglyConnectedBlocks*, *SubMatrix*, *SubVector*, *SumBasis*, *SylvesterMatrix*, *ToeplitzMatrix*, *Trace*, *Transpose*, *TridiagonalForm*, *UnitVector*, *VandermondeMatrix*, *VectorAdd*, *VectorAngle*, *VectorMatrixMultiply*, *VectorNorm*, *VectorScalarMultiply*, *ZeroMatrix*, *ZeroVector*, *Zip*]

Matrices are constructed in two different ways. The first involves using the `Matrix` command, `Matrix(m,n,L)`, where `m` and `n` denote the number of rows

and columns, respectively, and where L is a list of the matrix entries, reading across the rows. For example:

```
> A:=Matrix(2,3,[1,4,0,2,-3,7]);
```

$$A = \begin{bmatrix} 1 & 4 & 0 \\ 2 & -3 & 7 \end{bmatrix}$$

A second method for defining a matrix uses column vector notation. A column vector[1] in $\mathbb{R}^n$ can be entered as <a1,a2,a3,...,an>, where a1,...,an denote the entries of the vector. Matrices are then formed by adjoining vectors using the symbols <, |, and >. Here is the preceding matrix A entered in such a manner:

```
> v1:=<1,2>:
> v2:=<4,-3>:
> v3:=<0,7>:
> A:=<v1|v2|v3>;
```

$$A = \begin{bmatrix} 1 & 4 & 0 \\ 2 & -3 & 7 \end{bmatrix}$$

To "stack" vectors vertically, as opposed to aligning them in column vector form, replace each | separating two entries with a comma instead:

```
> v1:=<1,2>:
> v2:=<4,-3>:
> v3:=<0,7>:
> A:=<v1,v2,v3>;
```

$$A = \begin{bmatrix} 1 \\ 2 \\ 4 \\ -3 \\ 0 \\ 7 \end{bmatrix}$$

Once matrices are entered into Maple, various operations can be performed on them. To multiply matrices A and B use a period. A scalar multiplied by a matrix still requires the asterisk, however. The following worksheet illustrates the use of several LinearAlgebra package commands:

```
> restart;
> with(LinearAlgebra): # Load Linear Algebra Package.
```

[1]Unless specifically stated otherwise, we assume throughout this text that every vector is a column vector.

```
> v1:=<1,2,3>: v2:=<1,1,1>: v3:=<-1,1,3>: # Define three vectors,
  v1, v2, and v2.
> A:=<v1|v2|v3>; # Create a matrix A having v1, v2, and v3 as columns.
```

$$A := \begin{bmatrix} 1 & 1 & -1 \\ 2 & 1 & 1 \\ 3 & 1 & 3 \end{bmatrix}$$

```
> RowOperation(A, [1, 3], 2); # Replace the first row of A with
  the sum of the first row and twice the third row.
```

$$\begin{bmatrix} 7 & 3 & 5 \\ 2 & 1 & 1 \\ 3 & 1 & 3 \end{bmatrix}$$

```
> B:=Matrix(3,3,[1,1,2,2,4,-3,3,6,-5]): # Define a 3 by 3 matrix
  B using the Matrix command.
> A.B; # Compute the product matrix AB.
```

$$\begin{bmatrix} 0 & -1 & 4 \\ 7 & 12 & -4 \\ 14 & 25 & -12 \end{bmatrix}$$

```
> 2*A+IdentityMatrix(3); # Compute the sum of 2*A and the 3 by
  3 identity matrix.
```

$$\begin{bmatrix} 3 & 2 & -2 \\ 4 & 3 & 2 \\ 6 & 2 & 7 \end{bmatrix}$$

```
> x := Vector[row]([1, 2, 3]); # Define a row vector x.
```

$$[1, 2, 3]$$

```
> x.A; # Compute the vector-matrix product xA.
```

$$[14, 6, 10]$$

```
> Transpose(A); # Compute transpose matrix of A.
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ -1 & 1 & 3 \end{bmatrix}$$

```
> ReducedRowEchelonForm(A); # Compute reduced row echelon form
  of A.
```

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -3 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> b:=<2,1,0>; # Define a vector b.
```

$$b := \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

```
> LinearSolve(A,b,free=t); # Solve the matrix equation Ax=b, specifying
  any free variables be labeled using t with appropriate subscripts.
```

$$\begin{bmatrix} -1 - 2t_3 \\ 3 + 3t_3 \\ t_3 \end{bmatrix}$$

```
> b:=<b1,b2,b3>: # Define a new vector b having variable entries.
> M:=<A|b>; # Define the matrix M by augmenting A with b.
```

$$M := \begin{bmatrix} 1 & 1 & -1 & b1 \\ 2 & 1 & 1 & b2 \\ 3 & 1 & 3 & b3 \end{bmatrix}$$

```
> GaussianElimination(M); # Compute the row echelon form of M.
```

$$\begin{bmatrix} 1 & 1 & -1 & b1 \\ 0 & -1 & 3 & b2 - 2b1 \\ 0 & 0 & 0 & b3 + b1 - 2b2 \end{bmatrix}$$

```
> Determinant(B); # Compute the determinant of B.
```

$$-1$$

```
> MatrixInverse(B); # Compute inverse matrix of B.
```

$$\begin{bmatrix} 2 & -17 & 11 \\ -1 & 11 & -7 \\ 0 & 3 & -2 \end{bmatrix}$$

A word of warning is in order regarding Maple's row operation conventions. The `ReducedRowEchelon` command calculates the reduced row echelon form of a matrix. However, if the matrix contains variable entries, Maple will perform row operations that may include dividing by variable expressions, ignoring the fact that such expressions could be zero. For example, when `GaussianElimination(M)` is replaced by `ReducedRowEchelonForm(M)` in the preceding worksheet, the output changes to

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & -3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This output reflects Maple's assumption that the expression $b3 + b1 - 2b2$ is nonzero.

Finally, we point out two common issues that frequently arise in the context of arrays and matrices. First, for large arrays and matrices, Maple returns output in summary format. For example, an 11-by-2 matrix A is displayed as

$$A := \begin{bmatrix} 11 \times 2 \text{ Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Fortran\_order} \end{bmatrix}$$

To view the actual contents of the matrix, select the output, right-click, and select "Browse." Actual entries are displayed in table format, which can then be displayed in the worksheet and even exported to Excel. Second, Maple is capable of converting various data types from one form to another. Typical conversions used throughout this text include those converting arrays to matrices and vice versa. For example, `convert(A,Matrix)` converts an array, `A`, to a Matrix type, thereby permitting the use of matrix operations. The command `convert(B,array)` converts a Matrix object, `B`, to an array. A second frequently used conversion will be that from a list to a vector (or vice versa). For example, `convert(L,Vector)` converts a list, `L`, to a Vector. For a thorough list of data types and permissible conversions, type `?convert` at the command prompt.

C.6 Plot Structures with Maple

Maple is very versatile in its ability to create plots of different types. In this section we focus on four of these: functions and expressions, relations, regions defined by inequalities, and point plots.

Plotting Functions and Expressions

To plot a function of one variable, use the command `plot(`*`function`*`,`*`interval`*`,` *`options`*`)`; for a function of two variables, use `plot3d(`*`function`*`,`*`domain`*`,` *`options`*`)` instead. In each of these commands, the function should be given in expression form. For a function, `f` of the variable, `x`, this expression is `f(x)`, and for the function `f` of two variables, `x1` and `x2`, it is `f(x1,x2)`. In the former case, the plot interval takes the form `x=a..b`, where `a` and `b` are real numbers with `a` < `b`. For the function of two variables, the domain is expressed as a sequence of two intervals. Options can control plot color, range of output values, axes labeling, and numerous other plot features. For a complete list of

options, type either `?plot[options]` or `?plot3d[options]` at the command prompt. Here are two examples, which illustrate both the `plot` and `plot3d` commands and a few commonly used options:

```
> restart;
> f:=x->x+sin(x);
```

$$f := x \to x + \sin(x)$$

```
> plot(f(x),x=0..2*Pi,y=0..6,color=blue);
  #plot function in blue on interval [0,2*Pi], restricting output
  values to the interval [0,6].
```

The output is shown in Figure C.1.

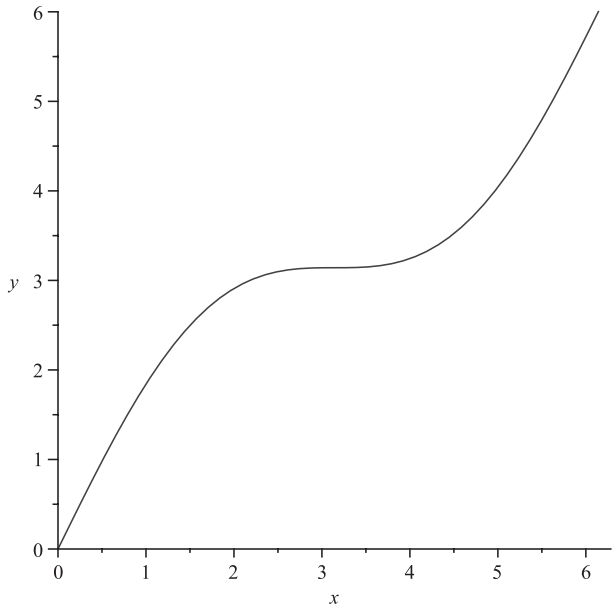


FIGURE C.1: Sample plot of a function of one variable.

```
> g:=(x1,x2)->x1*exp(-x1 ^2 -x2 ^2);
```

$$g := (x1, x2) \to x_1 e^{-x_1^2 - x_2^2}$$

```
> plot3d(g(x1,x2), x1=-2..2,x2=-2..2,color=red,style=wireframe,
  axes=framed);
  #plot function in red on specified rectangle using the
  wireframe style and a framed set of axes.
```

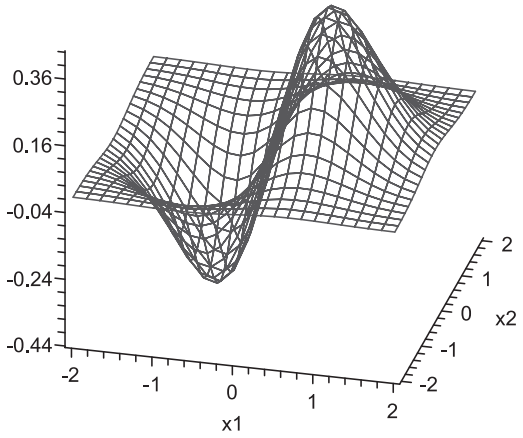


FIGURE C.2: Sample plot of a function of two variables.

The output is shown in Figure C.2.

To plot more than one function or expression on a single set of axes, enter the functions and/or expressions as a list within the plot command. Colors can be specified using a list in which each color matches the corresponding function or expression entry. Here is an example, in which the functions $f(x) = x + \sin(x)$, $y = x - 1$, and $y = x + 1$ are plotted on a single set of axes, in black, blue and red, respectively:

```
> plot([f(x),x-1,x+1],x=0..2*Pi,y=0..6,color=[black,blue,red]);
  # f, x-1, and x+1 are plotted black, blue, and red, respectively.
```

The output is shown in Figure C.3.

Relations and Other Important Plot Structures
Plot structures for functions are the simplest, but Maple is capable of producing numerous other types as well. Here we illustrate three examples, all of which require the `plots` package.

The first uses the `implicitplot` command to plot the relation $x = y^3 - y$.

```
> restart;
> with(plots):
```

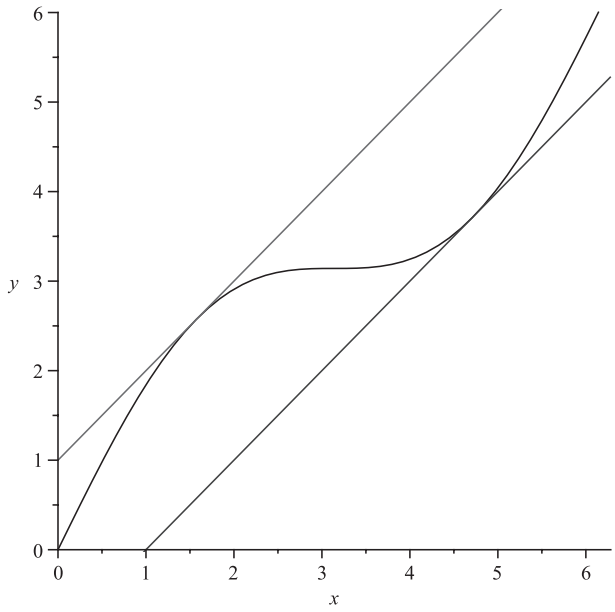


FIGURE C.3: Sample plot of three functions of a single variable.

```
> implicitplot(x=y^3-y, x=-4..4,y=-4..4,color=blue, grid=[50,50]);
  # Plot relation in blue on given rectangle;
  the rectangle is divided horizontally and vertically
  into a 50 by 50 grid in order to improve resolution .
```

The output is shown in Figure C.4.

The `pointplot` command plots a list of data points in $\mathbb{R}^2$, with each data point itself expressed as a list of two numbers.

```
> restart;
> with(plots):
> L:=[[1,2],[1,1],[0,.5],[-1,-1]];
```

$$L := [[1, 2], [1, 1], [0, .5], [-1, -1]]$$

```
> pointplot(L,color=blue,symbol=circle,symbolsize=20);
  # Plot points as blue circles using specified size.
```

The output is shown in Figure C.5.

To plot points in $\mathbb{R}^3$, one merely uses the `pointplot3d` command instead, where each point in the list `L` is expressed as a list of three numbers.

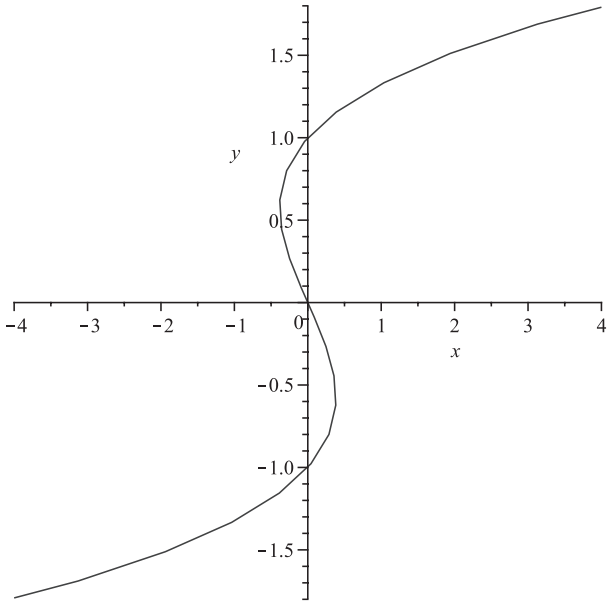


FIGURE C.4: Example of a plotted relation.

Finally, the `inequal` command is extremely useful for graphing regions in the $\mathbb{R}^2$ that satisfy a list of linear inequalities. Here is an example, which shades the region bounded by $x1 \leq 2$, $x2 \geq 1$, $x1 + x2 \leq 4$, and $x1 - x2 \geq -2$.

```
> restart;
> with(plots):
> inequal([x1<=2,x2>=1,x1+x2<=4,x1-x2>=-2],x1=-3..3,x2=0..5,
  optionsfeasible=(color=grey),
  optionsexcluded=(color=white),
  optionsclosed=(color=black));
  # Shade in grey the region of points satisfying all
  # inequalities in the list. The exterior and boundary
  # of the region are colored white and black, respectively.
```

The output is shown in Figure C.6.

Superimposing Plots

One of the most useful Maple tools for combining plot structures is the `display` command, which is located in the `plots` package. It allows one to superimpose a list of plot structures, all of whose members are plot structures in one of $\mathbb{R}^2$ or in $\mathbb{R}^3$. The general procedure for accomplishing this task first requires creating the individual plot structures themselves (e.g., `plot`,

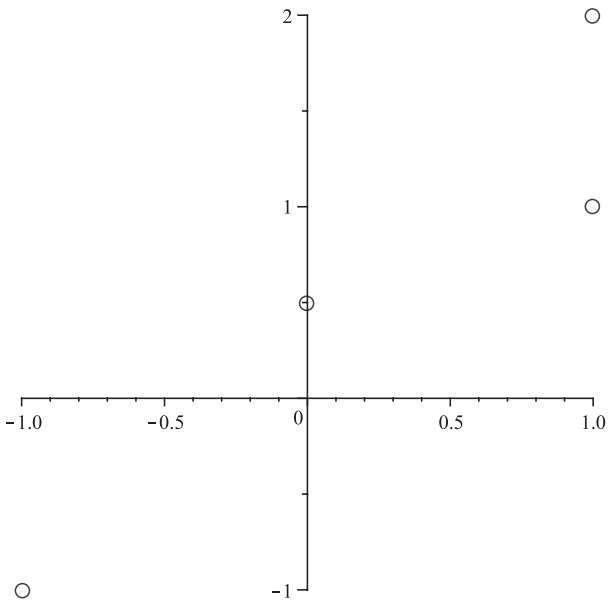


FIGURE C.5: Example of plotted points.

`implicitplot`, `pointplot`, etc.) as discussed previously. However, each plot structure is assigned a name and the command line performing the assignment ends with a colon so as to suppress output. Once the structures are all created in this manner, a list is formed using their respective names, and this list is used as the argument for the `display` command. Here is an example, which superimposes results from the preceding `pointplot` and `inequal` commands.

```
> restart;
> with(plots):
> L:=[[1,2],[1,1],[0,.5],[-1,-1]];
```

$$L := [[1, 2], [1, 1], [0, .5], [-1, -1]]$$

```
> G1:=pointplot(L,color=blue,symbol=circle,symbolsize=20):
> G2:=inequal([x1<=2,x2>=1,x1+x2<=4,x1-x2>=-2],x1=-3..3,x2=0..5,
  optionsfeasible=(color=grey), optionsexcluded=(color=white),
  optionsclosed=(color=black)):
> display([G1,G2]);
```

The output is shown in Figure C.7.

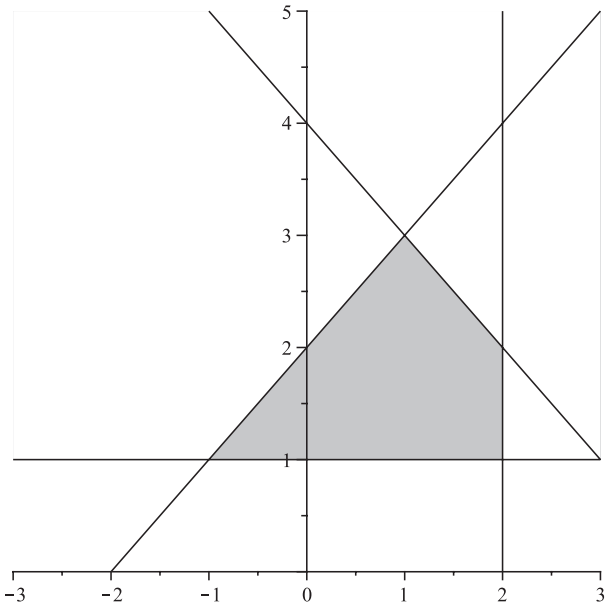


FIGURE C.6: Plot of region satisfying list of linear inequalities.

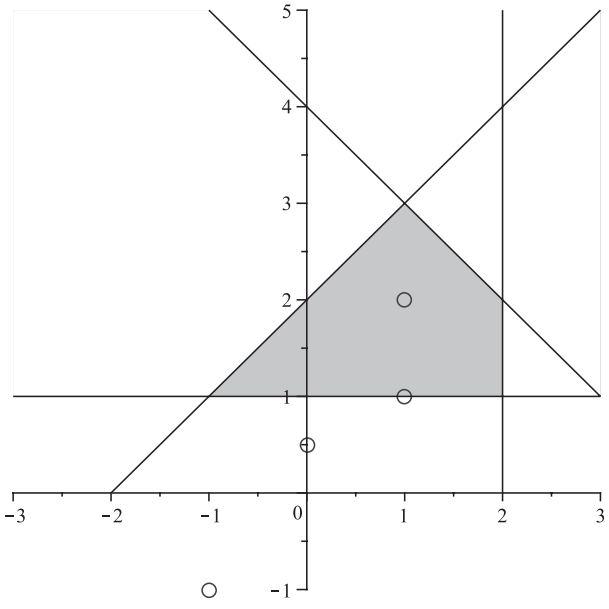


FIGURE C.7: Superposition of two plot structures.

Appendix D

Summary of Maple Commands

Here is a list of Maple commands used at various points in this text, organized by package name. Each command is accompanied by a brief explanation of its syntax as well as an example. To learn more about a particular command, type `?command name` at the command prompt. To load a particular package, type `with(package name)`.

Functions

1. `f:=x-> expression involving x:`

   Constructs a function named `f`, whose rule is governed by the given *expression*.

   Example:

   ```
   > f:=x->x^2 -x;
   ```

   $$f := x \to x^2 - x$$

   ```
   > f(3);
   ```

   $$6$$

   Functions of more than one variable are similarly defined.

   Example:

   ```
   > f:=(x1,x2)-> x1^2 -x2^2;
   ```

   $$f := (x_1, x_2) \to x_1^2 - x_2^2$$

   ```
   > f(2,1);
   ```

   $$3$$

   Piecewise-defined functions can be entered via the `piecewise` command, whose general form is

   `piecewise(cond1,expr1,...,condn, exprn,expr_otherwise),`

where each condition corresponds to the expression immediately following it. The conditions are expressed as inequalities and Boolean combinations thereof.

Example:

```
> f:=x->piecewise(x<=0,-x^2, x>0 and x<=2,x^2, 0);
```

$$f := x- > piecewise(x < 0, -x^2, x > 0 \text{ and } x \leq 2, x^2, 0)$$

Defines the function f, whose output equals $-x^2$ for negative inputs, x^2 for inputs between 0 and 2, inclusive, and 0 otherwise.

The `unapply` command defines functions as well.

Example:

```
> f:=unapply(x1^2 -x2^2,[x1,x2]);
```

$$f := (x_1, x_2) \rightarrow x_1^2 - x_2^2$$

Plot Structures

1. `plot(`*`expression`*`,`*`interval`*`,`*`options`*`);`

   Plots an *`expression`* of a single variable over the specified interval using prescribed *`options`*, such as graph color, etc.

   Example:

   ```
   > f:=x->x^2:
   > plot(f(x),x=-1..3,color=red);
   ```

   Plots the function $f(x) = x^2$ in red on the interval $[-1, 3]$

   For a complete list of `plot` options, type `?plot[options]` at the command prompt.

2. `plot3d(`*`expression`*`,`*`plot region`*`, `*`options`*`);`

   Plots an *`expression`* in two variables using specified *`plot region`* and *`options`*, which dictate surface type, color, etc.

   Example:

   ```
   > f:=(x1,x2)->x1*x2:
   > plot3d(f(x1,x2),x1=0..2,x2=-2..2,style=wireframe,color=blue);
   ```

   Plots the function $f(x_1, x_2) = x_1 x_2$ in blue, wireframe style on the region $0 \leq x_1 \leq 2$ and $-2 \leq x_2 \leq 2$.

For a complete list of `plot3d` options, type `?plot3d[options]` at the command prompt.

3. `implicitplot(`*`relation`*`,`*`options`*`);`

   Plots a *`relation`* in two variables using specified *`options`*, which dictate plot region, color, etc.

   Example:

   ```
   > implicitplot(x1^2-x2^2=1,x1=-5..5,x2=-5..5,color=green,
   thickness=3);
   ```

   Plots the relation $x_1^2 - x_2^2 = 1$ in green, thickness level 3 on the region $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$.

4. `contourplot(`*`relation`*`,`*`options`*`);`

   Creates a contour plot of an `expression` in two variables using specified `options`, which dictate plot region, color, contour values, etc.

   Example:

   ```
   > contourplot(x1^2-x2^2,x1=-5..5,x2=-5..5,color=blue,
   contours=[-2,-1,0,1,2]);
   ```

   Creates blue contour plot on the region $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$ of the expression $z = x_1^2 - x_2^2$ using z values of $z = -2, -1, 0, 1, 2$.

5. `inequal(`*`inequalities`*`,`*`options`*`);`

   Plots the region in the plane consisting of points that satisfy the given inequalities. The inequalities must be linear, allow for equality, and should be enclosed within brackets and separated by commas. The options dictate the plot region, the color of the region satisfying the inequalities, (`optionsfeasible`), the color of the boundary of the region, (`optionsclosed`), and the color of the points not satisfying at least one inequality, (`optionsexcluded`). This command is contained in the `plots` package.

   Example:

   ```
   > with(plots):
   > inequal([x1+x2 <= 3, x2 >= x1], x1 = 0 .. 5, x2 = 0 .. 5,
   optionsfeasible = (color = red), optionsclosed = (color =
   green), optionsexcluded = (color = yellow));
   ```

   Plots in red the set of points, (x_1, x_2), that satisfy the inequalities, $x_1 + x_2 \le 3$ and $x_2 \ge x_1$ and that belong to the plot region $0 \le x_1 \le 5$ and $0 \le x_2 \le 5$. The boundary of the region is colored green, and the points not satisfying at least one inequality are colored yellow.

This command also has coloring options for situations when at least one inequality is strict, i.e., involves $<$ or $>$ as opposed to $\leq$ or $\geq$.

6. `pointplot(`*`list_of_points`*`,`*`options`*`);`

   Plots the specified *`list_of_points`* using specified *`options`*. Each point should consist of two numbers, separated by commas and enclosed within brackets. Then these points themselves are separated by commas and enclosed within brackets again. Options permit specifying the point color, the point symbol, (asterisk, box, circle, cross, diagonalcross, diamond, point, solidbox, solidcircle, soliddiamond), and the size of this symbol, whose default value is 15. This command is contained in the `plots` package.

   Example:

   ```
   > with(plots):
   > pointplot([[-1,3],[0,0],[3,7]],symbol=box,symbolsize=20);
   ```

   Plots, as red boxes, the ordered pairs, $(-1, 3)$, $(0, 0)$, and $(3, 7)$. The size of the symbol is slightly larger than the default value.

   A similar command, `pointplot3d`, exists for pointing points in $\mathbb{R}^3$.

7. `display(`*`plot structures`*`,`*`options`*`);`

   Superimposes on a single set of axes, the given `plot structures`, all of which belong to $\mathbb{R}^2$ or all of which belong to $\mathbb{R}^3$, using the specified `options`. The plot structures themselves must be separated by commas and enclosed within brackets or braces.

   This command, which is located in the `plots` package, is frequently used to superimpose different plot structures that have been previously created and assigned different names.

   Example:

   ```
   > with(plots):
   > graph1:=pointplot([[-1,3],[0,0],[3,7]],symbol=box,symbolsize=20):
   > graph2:=inequal([x1+x2 <= 3, x2 >= x1], x1 = 0 .. 5, x2 =
     0 .. 5, optionsfeasible = (color = red), optionsclosed =
     (color = green), optionsexcluded = (color = yellow)):
   > display([graph1,graph2],axes=framed);
   ```

   Displays the superposition of the set of points specified by `graph1`, along with the region specified by `graph2`.

Basic Programming Structures

1. `if` *`condition`* `then` *`statement 1`* `else` *`statement 2`* `end if:`

   Performs *`statement 1`* provided *`condition`* is true and *`statement 2`* otherwise.

   Example:

```
> a:=5:
> if a > 2 then x:=5 else x:=0 end if:
> x;
```

5

2. `for` *`index`* `from` *`start`* `to` *`finish`* `by` *`change`* `do` *`statement`* `end do:`

   Perform *`task`* specified by *`statement`* using index value *`index`*, which varies from *`start`* to *`finish`* in increments of *`change`*.

   Example:

```
> for i from 0 to 20 by 5 do print(i) end do;
```

0

5

10

15

20

3. `while` *`condition`* `do` *`statement`* `end do:`

   Performs *`statement`* so long as *`condition`* is true.

   Example:

```
> a:=5:
> while a < 8 do a:=a+1: print(a): end do:
```

6

7

8

4. `proc(arguments) local variables1: global variables2: task: RETURN(output): end:`

   Performs programming procedure using input *arguments*. Local variables are internal to the procedure and cannot be accessed outside of it; global variables are accessible outside the procedure. The procedure returns *output*.

   Example:

```
> PowerCounter:=proc(number, value) local i:global n:
  n:=0: for i from 0 to 5 do if number^i <= value then n:=n+1:
  end if:
  RETURN(n):
  end:
> PowerCounter(2,33);
```

$$5$$

   Procedure for determining the number of powers of `number` that are less than or equal to `value`, where the powers vary from 0 to 5.

Arrays and Lists

1. `array(i..j,m..n);`

   Creates an array in two integer indices, with the first index ranging from `i` to `j` and the second from `m` to `n`.

   Example:

```
> A:=array(0..2,1..3);
```

$$array(0..2, 1..3, [])$$

   Array entries are frequently assigned values using looping structures.

   Example:

```
> A:=array(0..2,1..3);
> for i from 0 to 2 do for j from 1 to 3 do A[i,j]:=i+j:od:od:
```

   Creates an array, `A`, consisting of 9 entries, where the indices vary from 0 to 2 and 1 to 3, respectively. Each entry is the sum of the two corresponding indices. For example, $A[0,2] = 2$.

2. `[list-entries];`

   Creates an ordered list using *list-entries*. The values of *list-entries* must be separated by commas. Values in the list can be extracted using subscripts; the commands `nops` determines the size of the list, and `op` removes the outer brackets.

   Examples:

```
> L:=[1,2,4,8,16,32]:
> L[3];
```

$$4$$

```
> nops(L);
```

$$6$$

```
> op(L);
```

$$1, 2, 4, 8, 16, 32$$

```
> L:=[blue,red,black,green]:
> L[3];
```

$$black$$

```
> nops(L);
```

$$4$$

```
> op(L);
```

$$blue, red, black, green$$

Sequences, Sums, and Products

1. `seq(`*expression*`,`*index*`=`*start* `to` *finish*`);`

   Creates a sequence of values that results when the integer index value *index*, which varies between *start* and *finish*, is substituted into *expression*.

   Example:

```
> seq(3^i,i=-2..3);
```

$$\frac{1}{9}, \frac{1}{3}, 1, 3, 9, 27$$

2. `sum(`*`expression`*`,`*`index`*`=`*`start`* `to` *`finish`*`);`

   Computes a closed form for the sum that results when the integer *`index`* is substituted into *`expression`*, where *`index`* varies from *`start`* to *`finish`* and the resulting sequence of values are added together.

   Example:

   ```
   > sum((1/2)^i,i=0..infinity);
   ```

   $$2$$

3. `add(`*`expression`*`,`*`index`*`=`*`start`* `to` *`finish`*`);`

   Computes the explicit sum that results when the integer *`index`* is substituted into *`expression`*, where *`index`* varies from *`start`* to *`finish`* and the resulting sequence of values are added together.

   Example:

   ```
   > add(2^i,i=-2..6);
   ```

   $$\frac{511}{4}$$

4. `product(`*`expression`*`,`*`index`*`=`*`start`* `to` *`finish`*`);`

   Computes the explicit product that results when the integer *`index`* is substituted into *`expression`*, where *`index`* varies from *`start`* to *`finish`* and the resulting sequence of values are multiplied together.

   Example:

   ```
   > product(2^i,i=1..3);
   ```

   $$64$$

Linear Algebra

Commands in this section are contained in the `Linear Algebra` package.

1. `Vector(`*`entries`*`)`:

   Constructs a vector having components given by *`entries`*. The entries should be separated by commas and enclosed with brackets. By default, Maple assumes a vector is a column vector. To enter a row vector instead, type `Vector[row](`*`entries`*`)`.

   Examples:

   ```
   > with(LinearAlgebra):
   ```

```
> Vector([1,2]);
```

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

```
> Vector[row]([1,2]);
```

$$\begin{bmatrix} 1 & 2 \end{bmatrix}$$

Alternatively, a column vector can be entered using <, >, notation, e.g., `<1,2>`.

2. `Matrix(`*`m`*`,`*`n`*`,`*`entries`*`);`

   Constructs an *m*-by-*n* matrix having using prescribed *entries*, which are read by rows. The object *entries* should be separated by commas and enclosed with brackets.

   Example:

```
> with(LinearAlgebra):
> Matrix(2,3,[1,4,5,-3,2,5]);
```

$$\begin{bmatrix} 1 & 4 & 5 \\ -3 & 2 & 5 \end{bmatrix}$$

   Alternatively, a matrix can be entered by combining vectors and using the symbols < and >. Two vectors are augmented using | and stacked with a comma.

   Example:

```
> with(LinearAlgebra):
> v1:=<2,3>:
> v2:=<-4,0>:
> <v1|v2>;
```

$$\begin{bmatrix} 2 & -4 \\ 3 & 0 \end{bmatrix}$$

```
> <v1,v2>;
```

$$\begin{bmatrix} 2 \\ 3 \\ -4 \\ 0 \end{bmatrix}$$

3. `Norm(column vector,Euclidean);`

   Computes the Euclidean norm of `vector`, which is entered as a column vector using <, >, notation or a Matrix.

   Example:

```
> with(LinearAlgebra):
> u:=<3,4>:
> Norm(u,Euclidean);
                                5

> v:=Matrix(2,1,[3,4]):
> Norm(v,Euclidean);
                                5
```

   Note: Maple's `VectorCalculus` package also contains a `Norm` command, which accepts a row vector as its argument, as opposed to a column vector. Because of this fact, and for the sake of uniformity, when using both packages, we always load the `LinearAlgebra` package second.

4. `ZeroMatrix(m,n);`

   Constructs an m-by-n zero matrix; Similarly, `ZeroVector(m);` constructs a column vector with m entries.

   Example:

```
> with(LinearAlgebra):
> ZeroMatrix(2,3);
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

5. `IdentityMatrix(m);`

   Constructs an m-by-m identity matrix.

   Example:

```
> with(LinearAlgebra):
> IdentityMatrix(2);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

6. `UnitVector(m,n);`

   Constructs a column vector in $\mathbb{R}^n$ having an entry of 1 in component m and a zero in all other components.

   Example:

```
> with(LinearAlgebra):
> UnitVector(2,4);
```

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

7. `DiagonalMatrix(entries);`

   Constructs a square, diagonal matrix using *entries* along the diagonal. The entries should be separated by commas and enclosed with brackets.

   Example:

```
> with(LinearAlgebra):
> DiagonalMatrix([1,3,-5]);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -5 \end{bmatrix}$$

8. `Row(M,m);`

   Extracts, as a row vector, row *m* of the matrix *M*. Likewise, `Column(M,m)` extracts, as a column vector, column *n* of the matrix *M*.

   Example:

```
> with(LinearAlgebra):
> A:=Matrix(2,3,[1,4,5,-3,2,5]);
```

$$\begin{bmatrix} 1 & 4 & 5 \\ -3 & 2 & 5 \end{bmatrix}$$

```
> Row(A,2);
```

$$\begin{bmatrix} -3 & 2 & 5 \end{bmatrix}$$

9. `RowOperation(M,operation);`

   Performs a specified row operation, dictated by *operation*, on the matrix *M*. There are three such operations.

   `> RowOperation(M,j,k);`

   Multiplies row *j* of *M* by *k*.

   `> RowOperation(M,[i,j]);`

   Interchanges rows *i* and *j* of *M*.

   `> RowOperation(M,[i,j],k);`

   Adds *k* times row *j* to *i*.

Example:

```
> with(LinearAlgebra):
> A:=IdentityMatrix(3);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> RowOperation(A,3,4);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

```
> RowOperation(A,[1,3]);
```

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

```
> RowOperation(A,[1,3],-2);
```

Produces the matrix,

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Note: The option, *inplace=true*, replaces the value of the matrix with that obtained by performing the specified row operation. For example, `RowOperation(M,[1,2], inplace=true)` interchanges rows 1 and 2 of *M* and overwrites the value of *M* with this new matrix.

10. `Pivot(M,i,j);`

Pivots on the entry in row *i*, column *j* of the matrix *M*.

Example:

```
> with(LinearAlgebra):
> A:=Matrix(3, 3, [1, 4, 5, -3, 2, 5, 7, 0, -1]);
```

$$\begin{bmatrix} 1 & 4 & 5 \\ -3 & 2 & 5 \\ 7 & 0 & -1 \end{bmatrix}$$

```
> Pivot(A,1,3);
```

$$\begin{bmatrix} 1 & 4 & 5 \\ -4 & -2 & 0 \\ \frac{36}{5} & \frac{4}{5} & 0 \end{bmatrix}$$

11. `RowDimension(M);`

    Determines the row dimension of the matrix *M*. Likewise, `ColumnDimension(M);` determines the column dimension of *M*.

    Example:

```
> with(LinearAlgebra):
> A:=Matrix(2,3,[1,4,5,-3,2,5]):
> RowDimension(A)
```

$$2$$

```
> ColumnDimension(A)
```

$$3$$

12. `GaussianElimination(M);`

    Performs Gaussian elimination on the matrix *M* and produces the row echelon form.

13. `ReducedRowEchelonForm(M);`

    Computes the reduced row echelon form of *M*.

14. `LinearSolve(A,b,t);`

    Solve the matrix equation *Ax=b*, specifying any free variables be labeled using *t* along with appropriate subscripts.

    Example:

```
> A:=Matrix(3,3,[1,1,-1,2,1,1,3,1,3]);
```

$$A := \begin{bmatrix} 1 & 1 & -1 \\ 2 & 1 & 1 \\ 3 & 1 & 3 \end{bmatrix}$$

```
> b:=<2,1,0>;
```

$$b := \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

```
> LinearSolve(A,b,free=t);
```

$$\begin{bmatrix} -1 - 2t_3 \\ 3 + 3t_3 \\ t_3 \end{bmatrix}$$

15. `Determinant(M);`

    Computes the determinant of the matrix *M*.

16. `MatrixInverse(M);`

    Calculates the inverse of the matrix *M*. If the matrix is not invertible, an error message is returned.

17. `Eigenvalues(M);`:

    Calculates the eigenvalues of the matrix *M*.

18. `Eigenvectors(M);`

    Calculates the eigenvectors and corresponding eigenvalues of the matrix *M*. The eigenvalues are returned as a column vector, and the eigenvectors as corresponding columns in a matrix.

    Example:

```
> with(LinearAlgebra):
> A:=Matrix(3,3,[1, 0, 0, 2, 3, 0, 4, 5, 3]):
> Eigenvectors(A)
```

$$\begin{bmatrix}1\\3\\3\end{bmatrix}, \begin{bmatrix}2 & 0 & 0\\-2 & 0 & 0\\1 & 1 & 0\end{bmatrix}$$

    Thus, the matrix `A` has eigenvectors $\begin{bmatrix}2\\-2\\1\end{bmatrix}$ (corresponding eigenvalue of 1) and $\begin{bmatrix}0\\0\\1\end{bmatrix}$ (corresponding repeated eigenvalue of 3).

19. `IsDefinite(M,'query'=q);`

    Tests whether the matrix *M* is positive (semi-) definite, negative (semi-) definite, or indefinite. The returned value is *true* or *false*, and the parameter *q* specifies the matrix form to be determined. Choices for *q* consist of *'positive_definite'*, *'positive_semidefinite'*, *'negative_definite'*, *'negative_semidefinite'*, and `'indefinite'`.

    If the matrix has symbolic entries, the command returns conditions on these entries for which *q* is satisfied.

    Examples:

```
> with(LinearAlgebra):
> A:=Matrix(3,3,[1, 0, 0, 2, 3, 0, 4, 5, 3]):
> IsDefinite(A,'query'='positive_definite')
```

$$false$$

```
> A:=Matrix(3,3,[1, 0, 0, 2, 3, 0, 4, 5, x]):
```

```
> IsDefinite(A,'query'='positive_semidefinite');
```

$$0 \leq -\frac{33}{4} + 2x \text{ and } 0 \leq -\frac{33}{4} + 4x \text{ and } 0 \leq 4 + x$$

Importing Spreadsheet Data

This command is contained in the `Exceltools` package.

1. `Import(‘‘`*`directory filename.xls`*`’’, ’’sheetname’’,’’cellrange’’)`:

   Imports cells *`cellrange`*, from worksheet, *`sheetname`*, of the Excel file, *`filename.xls`*, located in *`directory`*. All command arguments are contained in quotation marks, and the data itself is imported into an array, which can then be converted to a `Matrix` and/or `Vector` structure.

   Example:

```
> with(ExcelTools):
> A:=convert(Import(''c
  file.xls'', ''sheet1'', ''A1:D5''),Matrix):
> B:=convert(Import(''c
  file.xls'', ''sheet1'', ''E1:E5''),Vector):
```

Basic Statistics Commands

Commands in this section are contained in the `Statistics` package.

1. `Mean(`*`L`*`)`:

   Computes the mean value of the entries in the list, *`L`*.

2. `Variance(`*`L`*`)`:

   Computes the variance of the entries in the list, *`L`*.

   Example:

```
> with(Statistics):
> L:=[1,2,3,4,5]:
> Mean(L);
```

$$3$$

```
> Variance(L);
```

$$2.5$$

Linear and Nonlinear Optimization

Commands in this section are contained in the `Optimization` package.

1. `LPSolve(`*`expression`*`, `*`constraints`*`,`*`options`*`)`

   Solves the linear programming problem consisting of objective, *`expression`*, subject to the entries of *`constraints`*. The quantity *`expression`* must be linear in the decision variables, and *`entries`* must consist of linear inequalities in the decision variables, separated by commas and enclosed within brackets. The command permits a variety of *`options`*, which can specify whether the objective is to be minimized or maximized (the default goal is minimization), can dictate sign restrictions on decision variables, and can require one or more decision variables to be integer-valued in the solution. The output of the command is a list, with the first entry given by the optimal objective value and the second as a vector specifying the corresponding decision variable values.

   Examples:

```
> with(Optimization):
> LPSolve(-4*x1-5*x2, [x1+2*x2<=6, 5*x1+4*x2<=20],assume=
  nonnegative);
```

$$[-19, [x_1 = 2.6666, x_2 = 1.6666]]$$

```
> LPSolve(-7*x1+2*x2, [4*x1-12*x2 <= 20, -x1+3*x2 <= 3],
  assume = 'nonnegative', maximize);
```

$$[2, [x_1 = 0, x_2 = 1]]$$

```
> LPSolve(3*x1-2*x2, [x1-2*x2 <= 5, -x1+3*x2 <= 3, x1 >= 2],
  assume = integer);
```

$$[4, [x_1 = 2, x_2 = 1]]$$

```
> LPSolve(x1-2*x2, [x1-2*x2 <= 5, -x1+3*x2 <= 3, x1 >= 0],
  assume = binary);
```

$$[-2, [x_1 = 0, x_2 = 1]]$$

   The `LPSolve` command has several other options in addition to those given above.

2. `LPSolve(`*`c`*`, [`*`A`*`,`*`b`*`],`*`options`*`);`

   The matrix form of the `LPSolve` command in which the LP consists of minimizing *`c`*`x` subject to *`A`*`x` $\leq$ *`b`* and the prescribed *`options`*. The quantities *`c`* and *`b`* are vectors, and *`A`* is a matrix whose dimensions are such that the matrix vector product, *`A`*`x`, is well defined and has the same number of entries as *`b`*. The output of the command is a list, with

the first entry given by the optimal objective value and the second as a vector representing **x**

Example:

```
> with(Optimization):
> c:=Vector[row]([-7,2]);
```

$$c = [-7, 2]$$

```
> b:=<20,3>;
```

$$b = \begin{bmatrix} 20 \\ 3 \end{bmatrix}$$

```
> A:=Matrix(2,2,[4,-12,-1,3]);
```

$$A = \begin{bmatrix} 4 & -12 \\ -1 & 3 \end{bmatrix}$$

```
> LPSolve(c, [A,b], assume = nonnegative,'maximize');
```

$$\left[2, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right]$$

3. `NLPSolve(`*`expression`*`, `*`constraints`*`,`*`options`*`)`

   Similar to the `LPSolve` command, `NLPSolve` minimizes *`expression`*, subject to *`constraints`*, which consists of a list of inequalities. Both *`expression`* and any particular constraint, are permitted to be nonlinear in the decision variables. The advanced numeric method used by Maple to execute this command generally returns a local optimal solution. However, if the problem is convex, this local optimal solution is a global optimal solution as well.

   Example:

```
> with(Optimization):
> NLPSolve(x1^2+x2^2+3*x2, [x1^2+x2^2 <= 3, x1-x2 <= 4], assume
  = nonnegative, 'maximize');
```

$$[8.19615242273757792, [x_1 = 0, x_2 = 1.73205080757366470]]$$

4. `QPSolve(`*`quadratic expression`*`, `*`constraints`*`,`*`options`*`)`

   A special case of the `NLPSolve` command is `QPSolve`, which solves quadratic programming problems. In this situation, the objective, *`quadratic expression`*, is quadratic in the decision variables. Each entry of *`constraints`* takes the form of a linear or nonlinear inequality.

Example:

```
> with(Optimization):
> QPSolve(x1^2+x2^2+3*x2, [x1+x2 <= 3, x1-x2 <= 4]);
```

$$[-2.25, [x_1 = 0, x_2 = -1.5]]$$

5. `QPSolve([p,Q],[C,d,A,b],options);`

   This is the matrix form of the QPSolve command, which solves the quadratic programming problem of minimizing $\frac{1}{2}\mathbf{x}^t Q\mathbf{x} + p^t\mathbf{x}$ subject to the linear constraints, $A\mathbf{x} = b$ and $C\mathbf{x} \leq d$. Note that p, b, and d must be entered as vectors and not as matrices. In addition, if only C and d are listed within the brackets, Maple assumes they correspond to the linear inequalities. If the problem involves equality constraints only, constraints are given by `[NoUserValue,NoUserValue,A,b]`.

   Examples:

```
> with(Optimization):
> Q:=Matrix(2,2,[2,1,1,3]):
> p:=<5,-3>:
> C:=Matrix(2,2,[1,0,0,3]):
> d:=<3,0>:
> A:=Matrix(2,2,[4,2,2,1]):
> b:=<6,3>:
> QPSolve([p,Q],[C,d,A,b]);
```

$$\left[9.75, \begin{bmatrix} 1.5 & 0 \end{bmatrix}\right]$$

```
> QPSolve([p,Q],[C,d]);
```

$$\left[-6.25, \begin{bmatrix} -2.5 & 0 \end{bmatrix}\right]$$

```
> QPSolve([p,Q],[NoUserValue,NoUserValue,A,b]);
```

$$\left[3.70, \begin{bmatrix} .4 & 2.2 \end{bmatrix}\right]$$

Vector Calculus

Commands in this section are contained in the `VectorCalculus` package.

1. `Gradient(expression, variables)`

   Calculates the gradient vector of the multivariable *expression* with respect to the variables prescribed by `variables`. The quantity *variables* consists of comma separated variables enclosed within brackets. The output of the command is expressed in terms of unit vectors corresponding to each variable. Each such vector takes the form $\mathbf{e}_{\text{variable}}$

   Example:

```
> with(VectorCalculus):
> Gradient(x1^2+2*x1*x2,[x1,x2]);
```

$$(2x_1 + 2x_2)\bar{e}_{x1} + 2x_1\bar{e}_{x2}$$

2. `Jacobian(`*`expressions, variables`*`)`

   Calculates the Jacobian matrix of the multivariable *`expressions`* with respect to the variables prescribed by *`variables`*. The quantity *`expressions`* consists of comma-separated expressions enclosed within brackets, and *`variables`* consists of comma separated variables enclosed within brackets. The output of the command is expressed as a matrix.

   Example:

```
> with(VectorCalculus):
> Jacobian([x1^2+2x1*x2,4x2^2-3x1*x2] ,[x1,x2]);
```

$$\begin{bmatrix} 2x_1 + 2x_2 & 2x_1 \\ -3x_2 & 8x_2 - 3x_1 \end{bmatrix}$$

3. `Hessian(`*`expression, variables`*`)`

   Calculates the Hessian matrix of the multivariable `expression` with respect to the variables prescribed by `variables`. The quantity `variables` consists of comma separated variables enclosed within brackets.

   Example:

```
> with(VectorCalculus):
> Hessian(x1^2+2*x1*x2,[x1,x2]);
```

$$\begin{bmatrix} 2 & 2 \\ 2 & 0 \end{bmatrix}$$

Bibliography

[1] Apte, A., Apte, U., Beatty, R., Sarkar, I., and Semple, J., The Impact of Check Sequencing on NSF (Not-Sufficient Funds) Fees, *Interfaces*, **34** (2004), 97-105.

[2] Avriel, M., *Nonlinear Programming*, Dover Publications, Mineola, NY, 2003.

[3] Bartlett, A., Chartier, T., Langville, A., and Rankin, T., An Integer Programming Model for the Sudoku Problem, *The Journal of Online Mathematics and its Applications*, **9** (2008).

[4] Bazaraa, M., Sherali, H., and Shetty, C., *Nonlinear Programming: Theory and Applications*, John Wiley and Sons, Hoboken, 2006.

[5] Belovsky, G., Herbivore Optimal Foraging: A Comparative Test of Three Models, *The American Naturalist*, **124** (1984), 97-115.

[6] Bernstein, D., *Matrix Mathematics*, Princeton University Press, Princeton, 2005.

[7] Bevington, P., and Robinson, D., *Data Analysis and Error Analysis for the Physical Sciences*, McGraw-Hill, New York, 2002.

[8] Blandford, D., Boisvert, R., and Charles, C., Import Substitution for Livestock Feed in the Caribbean Community, *American Journal of Agricultural Economics*, **64**, (1982), 70-79.

[9] Bouskila, A., A Habitat Selection Game of Interactions Between Rodents and Their Predators, *Annales Zoologici Fennici*, **38** (2001), 55-70.

[10] Dantzig, G., and Thapa, M., *Linear Programming 2: Theory and Extensions*, Springer, New York, 2003.

[11] DeWitt, C., Lasdon, L., Waren, A., Brenner, D., and Melhem., S., OMEGA: An Improved Gasoline Blending System for Texaco, *Interfaces*, **19** (1989), 85-101.

[12] Duncan, I.B., and Noble, B.M., The Allocation of Specialties to Hospitals in a Health District, *The Journal of the Operational Research Society*, **30** (1979), 953-964.

[13] Floudas, C.A., Pardalos, P.M., (Eds.) *Recent Advances in Global Optimization*, Princeton University Press, Princeton, 1992.

[14] Heller, I., and Tompkins, C.B., An Extension of a Theorem of Dantzig's, in, *Linear Inequalities and Related Systems*, Kuhn, H.W. and Tucker, A.W. (Eds.) Princeton University Press, Princeton, 1956, 247-254.

[15] Horn, R., and Johnson, C., *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

[16] Horst, R., and Tuy, H., *Global Optimization - Deterministic Approaches*, Third Edition, Springer, Berlin, 1996.

[17] Jarvis, J., Rardin, R., Unger, V., Moore, R., Schimpler, C., Optimal Design of Regional Wastewater Systems: A Fixed-Charge Network Flow Model, *Operations Research*, **26** (1978), 538-550.

[18] Karmarkar, N., A New Polynomial-Time Algorithm for Linear Programming, *Combinatorica*, **4** (1984), 373-395.

[19] Klee, V., and Minty, G.J., How Good is the Simplex Method?, in *Inequalities* III, Shisha, O., (Ed.), Academic Press, New York, 1972, 159-175.

[20] Ladany, S., Optimization of Pentathlon Training Plans, *Management Science*, **21**, (1975), 1144-1155.

[21] Lay, D., *Linear Algebra*, Third Edition, Addison-Wesley, New York, 2006.

[22] Letavec, C., and Ruggiero, J., The n-Queens Problem, *INFORMS Transactions on Education*, **2** (2002), 101-103.

[23] Levenberg, K., A Method for Solution of Certain Non-linear Problems in Least-squares, *Quarterly of Applied Mathematics*, **2** (1944), 164-168.

[24] Luenberger, D., *Linear and Nonlinear Programming*, Second Edition, Springer, New York, 2003.

[25] Machol, R., An Application of the Assignment Problem, *Operations Research*, **18**, 1970, 585-586.

[26] Mangasarian, O., Street, W., and Wolberg, W., Breast Cancer Diagnonsis and Prognosis via Linear Programming, Mathematical Programming Technical Reports 94-10, Madison, WI, 1994.

[27] Mangasarian, O., Equilibrium Points of Bimatrix Games, *Journal of the Society for Industrial and Applied Mathematics*, **12** (1963), 778-780.

[28] Marquardt, D., An Algorithm for Least Squares Estimation of Parameters, *Journal of the Society for Industrial and Applied Mathematics*, **11** (1963), 431-441.

[29] Marsden, J., and Tromba., A., *Vector Calculus*, Fifth Edition, W.H. Freeman and Company, New York, 2003.

[30] Marshall, K., and Suurballe, J., A Note on Cycling in the Simplex Algorithm, *Naval Research Logistics Quarterly*, **16** (1969) 121-137.

[31] Maynard, J., A Linear Programming Model for Scheduling Prison Guards, UMAP Module 272, The Consortium for Mathematics and its Applications, Birkhauser, Boston, 1980.

[32] McIntyre, L, Using Cigarette Data for an Introduction to Multiple Regression, *Journal of Statistics Education* (online), **2** (1994).

[33] Moreb, A., and Bafail, A., A Linear Programming Model Combining Land Leveling and Transportation Problems, *The Journal of the Operational Research Society*, **45** (1994), 1418-1424.

[34] Nash, J., Non-Cooperative Games, *The Annals of Mathematics*, **54** (1951), 286-295.

[35] Neumaier, A., *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990.

[36] Nocedal, J., and Wright, S., *Numerical Optimization*, Springer, New York, 2000.

[37] Pendegraft, N., Lego of my Simplex, *ORMS Today* (online), **24** (1997).

[38] Penrose, K., Nelson, A., and Fisher, G., Generalized Body Composition Prediction Equation for Men Using Simple Measurement Techniques, *Medicine and Science in Sports and Exercise*, **17** (1985), 189.

[39] Pintér, J.D., *Global Optimization in Action*, Kluwer Academic Publishers, Dordrecht, 1996.

[40] Pintér, J.D., *Optima*, Mathematical Programming Society Newsletter, **52** (1996).

[41] Polak, E., *Optimization: Algorithms and Consistent Approximations*, Springer, New York, 1997.

[42] Reed, H.S., and Holland, R.H., The Growth Rate of an Annual Plant *Helianthus*, *Proceedings of the National Academy of Sciences*, **5** (1919), 135-144.

[43] Schuster, E., and Allen, S., Raw Material Management at Welch's Inc., *Interfaces*, **28** (1998), 13-24.

[44] Sharp, J., Snyder, J., and Green, H., A Decomposition Algorithm for Solving the Multifacility Production Problem with Nonlinear Production Costs, *Econometrica*, **38** (1970), 490-506.

[45] Straffin, P., *Applications of Calculus*, The Mathematical Association of America, Washington D.C., 1996.

[46] Von Neumann, J., and Morgenstern, O., *Theory of Games and Economic Behavior*, Third Edition, Princeton University Press, Princeton, 1980.

[47] Yamashita, N., Fukushima, M., On the Rate of Convergence of the Levenberg-Marquardt Method, *Computing*, (Supplement), **15** (2001), 239-249.

[48] Zoutendijk, G., Nonlinear Programming, Computational Methods, in *Integer and Nonlinear Programming*, Abadie, J., (Ed.), North Holland, Amsterdam, 1970.

Index