The Open University

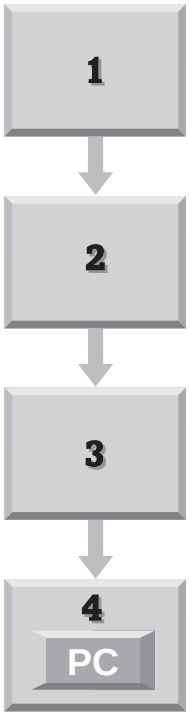
M373

Optimization

Linear programming – the basic ideas

Study guide

The four sections of this unit should take you roughly equal amounts of time to work through.



Many of the linear algebra techniques that you met in *Units I.2* and *I.3* of this course and may have met in other courses (such as MST209 and M208) are used throughout this unit. You will also find some of the modelling ideas from *Unit I.5* useful.

A few theorems are stated, without proof, in Sections 2 to 4.

There are computer activities at the end of Section 4.

Note to reader
Mathematical/statistical content at the Open University is usually provided to students in printed books, with PDFs of the same online. This format ensures that mathematical notation is presented accurately and clearly. The PDF of this extract thus shows the content exactly as it would be seen by an Open University student. Please note that the PDF may contain references to other parts of the module and/or to software or audio-visual components of the module. Regrettably mathematical and statistical content in PDF files is unlikely to be accessible using a screenreader, and some OpenLearn units may have PDF files that are not searchable. You may need additional help to read these documents.

This publication forms part of an Open University course. Details of this and other Open University courses can be obtained from the Student Registration and Enquiry Service, The Open University, PO Box 197, Milton Keynes MK7 6BJ, United Kingdom (tel. +44 (0)845 300 6090, email general-enquiries@open.ac.uk).

Alternatively, you may visit the Open University website at www.open.ac.uk where you can learn more about the wide range of courses and packs offered at all levels by The Open University.

To purchase a selection of Open University course materials visit www.ouw.co.uk, or contact Open University Worldwide, Walton Hall, Milton Keynes MK7 6AA, United Kingdom, for a brochure (tel. +44 (0)1908 858793, fax +44 (0)1908 858787, email ouw-customer-services@open.ac.uk).

The Open University, Walton Hall, Milton Keynes, MK7 6AA.

First published 2002. Second edition 2009.

Edited, designed and typeset by The Open University, using the Open University TEX System.

Printed and bound in the United Kingdom by Cambrian Printers Limited, Aberystwyth.

The paper used in this publication is procured from forests independently certified to the level of Forest Stewardship Council (FSC) principles and criteria. Chain of custody certification allows the tracing of this paper back to specific forest-management units (see www.fsc.org).

ISBN 978 0 7492 4976 2

2.1

Contents

Introduction

This block and the next are concerned with *optimization* problems. As was stated in the Introduction to *Unit I.1*, an **optimization** problem, when modelled mathematically, usually results in a real function whose *optimum* value, i.e. whose *maximum* or *minimum* value, is sought.

The techniques for arriving at a solution to an optimization problem are collectively known as **mathematical programming**. Such techniques were developed during the Second World War to help solve the problems of finding the most economical and effective strategies for deploying aircraft, submarines, troops and so on. Interest in the techniques continued after the war, when they were applied to industrial and government planning, and flourished with the development of computers, which enabled large-scale problems to be solved with comparative ease.

Note that the word 'programming' here means the formulation of a *programme*, or plan, for decision-making, and is not to be confused with computer programming, where the end result is a *program*.

In the Introduction to *Unit I.1*, it was noted that many optimization problems, when modelled, consist not only of a function to be optimized but also of one or more constraints on the values the variables can take. Such problems are known as **constrained optimization** problems, and arise frequently in *operational research*, for example, when management decisions have to be made regarding the optimal deployment of resources. For instance, a manager may want to plan production so as to *maximize* profit subject to *constraints* on manufacturing capacity and on the quality of the product.

In this block, we focus on constrained optimization problems. Block III deals with both constrained and **unconstrained optimization** problems (i.e. problems where there are no constraints on the values of the variables).

In fact, in this block, we only consider constrained optimization problems where the function to be optimized and the constraints can all be expressed as *linear* combinations of the variables. In this unit and the next we consider **linear programming** problems, where the variables are allowed to take on any real value (subject to the constraints). Here in *Unit II.1* we concentrate on the formulation and solution of small linear programming problems, and in *Unit II.2* we extend the methods to large problems and their solution on a computer. In *Unit II.3*, we consider **integer programming** problems, where some of the variables are restricted to integer values. *Unit II.4* looks at applications of linear and integer programming.

Section 1 of this unit deals with the *formulation* of linear programming models, describing how mathematical models of suitable real-world problems can be constructed. In Section 2 we look at graphical representations of two-dimensional models, consider some theoretical implications and examine the graphical solution of such models. In Section 3 we introduce the *simplex method* for solving linear programming models, and in Section 4 the method is formalized in matrix notation.

1 Formulation

Before a real-world problem can even be recognized as a candidate for linear programming, much preliminary work is required. This work essentially comprises Stage 1 of the mathematical modelling process described in *Unit I.5*, though parts of some of the later stages may need to be undertaken at this point as well. The result is a precise statement of the problem, usually accompanied by any necessary data and appropriate assumptions. In this block you will always be presented with problem statements containing all the required information necessary for creating a linear programming model, but you should be aware that in practice problems rarely appear in this form initially.

Remember that the mathematical modelling process described in *Unit I.5* is only a guideline and not a prescription.

In this section you will see how linear programming models can be formulated, given a suitable problem statement. The general form of such models is discussed in Subsection 1.1, while Subsections 1.2 and 1.3 go on to look at forms more suited to the methods of solution and theoretical results discussed later in this unit and in the next.

1.1 Some simple examples

In this subsection we shall see how to formulate a linear programming model, given a complete and precise statement of the problem, including its purpose, any simplifying assumptions and any necessary data. In the terminology of *Unit I.5*, we shall create the model once its purpose has been specified and the assumptions have been stated. We begin with an example.

Example 1.1

A small cooperative craft workshop makes two types of table: a standard rectangular table and a de luxe circular table. The market can absorb as many of either type of table as the workshop can produce, and so we can assume unlimited demand. Each type of table is made from the same wood and, once the wood has been cut, each table has to go through three processes: joinery, prefinishing and final finishing (in that order). Sufficient cut wood is always available. Each rectangular table takes 2 hours for joinery, 40 minutes for prefinishing and 5 hours 20 minutes for final finishing. Each circular table requires 3 hours for joinery, 2 hours for prefinishing and 4 hours for final finishing. The workshop employs five joiners, two sanders and eight polishers. The joiners each work a fixed six-hour day, while the sanders and polishers each work a fixed eight-hour day on the prefinishing and final finishing respectively. No overtime is worked, and full six-hour or eight-hour days are worked by each employee irrespective of whether there is work for that employee to do. All running costs, including wages, are fixed. The cooperative sells each rectangular table for £120 and each circular table for £150. How many of each type of table should it produce each day in order to maximize its profit? ■

We shall return to the question of cutting the wood in *Unit II.2*.

We shall ignore any difficulties caused by the three processes (joinery, prefinishing and final finishing) having to be performed sequentially, and assume that each process has a backlog of work sufficient to even out any problems.

This example is obviously simplified but it includes features common to many mathematical programming problems: there is a quantity (the profit) to be maximized and there are constraints (the available person-hours) on each process involved in making the tables. We want to formulate a mathematical model of the problem, where the purpose of the model is to decide how many of each type of table should be produced each day in order to maximize profit.

Having specified the purpose of the model, the mathematical modelling process of *Unit I.5* now suggests that we should state the assumptions. For the example, we can identify the following assumptions from the given description.

(a) There is unlimited demand for both types of table.

(b) Each table is made from the same type of wood.

(c) Sufficient cut wood is always available.

(d) No overtime is worked.

(e) Full days are worked irrespective of whether there is work to do.

(f) All running costs, including wages, are fixed.

Although we could derive such a list of assumptions for every example we discuss in this block, we shall not generally do so. Instead, we shall leave it to you to ascertain the assumptions from the given description, and move directly from a specification of the purpose of the model to its formulation, based on the description (which will include the necessary assumptions).

In formulating the model for Example 1.1, the following definitions will be useful.

Definitions

The quantity to be optimized in a mathematical programming problem is known as the **objective function**, and the **objective** is to optimize this function.

Formulation of Example 1.1

The first step in formulating a linear programming model is to identify the objective and the objective function. In this example, we want to maximize the daily profit, so we could take maximizing the daily profit as the objective and the daily profit as the objective function. However, the daily profit is equal to the daily income from selling tables less the *fixed* daily running costs (including the cost of wood and wages). So, since the running costs are fixed, maximizing daily profit is equivalent to maximizing daily income. As omitting the fixed running costs will simplify the model, we shall therefore take the objective to be to maximize daily income and the objective function to be the daily income.

Daily running costs (including the cost of wood and wages) are fixed by Assumptions (b), (c), (d), (e) and (f).

It is common practice in linear and integer programming to omit any constant terms, such as fixed running costs, from the objective function. This topic is discussed briefly at the end of the subsection.

In the terminology of *Unit I.5*, the objective function can be thought of as a (dependent) variable of the problem that we need to relate to the other (independent) variables. As such, we need to specify units for its measurement. Here, given the data, a suitable unit of measurement is pounds.

The second step is to identify the variables, other than the objective function, and to specify their units of measurement. In this example, the only variables are the numbers of tables of each type made per day.

The third step is to identify the constraints and parameters. In this example, the only constraints are the numbers of person-hours available for joinery, prefinishing and final finishing each day. The parameters are the time it takes for each process for each type of table, the hours available for each process each day, and the selling prices of the tables.

Assumptions (a) and (c) tell us that there are no constraints on the number of tables that can be sold or on the availability of cut wood.

The fourth step is to assign algebraic symbols to the objective function, the variables and, if necessary, the parameters. It is usual in linear programming to use z for the objective function and x_1, x_2, ... for the variables. So, in this case, we have:

z the daily income, in pounds;
x_1 the number of rectangular tables made per day;
x_2 the number of circular tables made per day.

In *Unit I.5*, you saw that it is desirable to assign symbols to parameters so that the same mathematical model can be used in a variety of circumstances by giving different values to the parameters. This is also true of linear programming models, and a general formulation using symbols for the parameters is given below. One approach to formulating a linear programming model for a specific situation would be to assign appropriate values to the parameters in the general formulation. However, it is common practice in linear and integer programming simply to bear the general formulation in mind while modelling a particular situation, and to use the numerical values of the parameters during the formulation process, resulting in a particular model for the given problem rather than a more general model that can be used for variations on the given specification. One reason for this is that the methods of solution used for linear and integer programming models can only solve models with numerical values for the parameters. We shall therefore adopt this approach here, so that, usually, instead of assigning symbols to the parameters, we shall state their numerical values in a table that relates them to the variables, the objective function and the constraints, as in Table 1.1.

Variations on the given specification can be modelled by changing the parameter values.

Table 1.1

	rectangular table	circular table	upper limit (per day)
income (£)	120	150	—
joinery (hours)	2	3	30
prefinishing (hours)	$\frac{2}{3}$	2	16
final finishing (hours)	$5\frac{1}{3}$	4	64

The upper limit on the number of hours per day for each process is calculated by multiplying the number of workers available for each process by the number of hours each works per day.

The next step is to derive algebraic relationships for the objective function and the constraints. Tables such as Table 1.1 are very useful in this respect. Before deriving these relationships, it is important to remember that they must be *linear* (i.e. they must be expressed using *linear* combinations of the variables), or else we will not have a *linear* programming model. In this case, there is a clear linear relationship between the daily income and the numbers of tables sold, given by

If the relationships are not linear then further simplifying assumptions may be needed or a different sort of model will need to be derived.

$$z = 120x_1 + 150x_2.$$

This is the objective function, and, since we wish to maximize it, the objective is

Notice how the objective function can be read off directly from the first row of Table 1.1.

$$\text{maximize } z = 120x_1 + 150x_2.$$

The numbers of hours spent daily on each of joinery, prefinishing and final finishing are given, using Table 1.1, by the simple linear expressions

$$\begin{aligned} &2x_1 + 3x_2 && \text{(joinery)},\\ &\tfrac{2}{3}x_1 + 2x_2 && \text{(prefinishing)},\\ &5\tfrac{1}{3}x_1 + 4x_2 && \text{(final finishing)}. \end{aligned}$$

The upper limits on the numbers of hours available for each of these processes each day can then be combined with these expressions to give the following linear constraints.

$$\begin{aligned} 2x_1 + 3x_2 &\leqslant 30 \\ \tfrac{2}{3}x_1 + 2x_2 &\leqslant 16 \\ 5\tfrac{1}{3}x_1 + 4x_2 &\leqslant 64 \end{aligned}$$

Notice how these can be read off directly from the last three rows of Table 1.1.

Finally we must, as so often in mathematics, state the obvious: the cooperative cannot make a negative number of tables. So we must also include the constraints:

$$x_1 \geqslant 0 \quad \text{and} \quad x_2 \geqslant 0.$$

We can write the objective and constraints succinctly as follows.

$$\text{maximize } z = 120x_1 + 150x_2 \tag{1.1}$$

subject to

$$\left.\begin{aligned} 2x_1 + 3x_2 &\leqslant 30 \quad \text{(joinery)} \\ \tfrac{2}{3}x_1 + 2x_2 &\leqslant 16 \quad \text{(prefinishing)} \\ 5\tfrac{1}{3}x_1 + 4x_2 &\leqslant 64 \quad \text{(final finishing)} \end{aligned}\right\} \tag{1.2}$$

$$x_1, x_2 \geqslant 0 \tag{1.3}$$

This completes the formulation of the problem as a linear programming model. ■

All **linear programming models** are of a form similar to that obtained in Example 1.1. They consist of a linear objective function to be maximized or minimized subject to linear constraints, which may be inequalities or equalities. The important point is that both the objective function and the constraints are *linear*, i.e. are expressed using linear combinations of the variables, with no powers, products or other functions of the variables permitted. Also, the variables are always restricted to non-negative values, i.e. the model must include a **non-negativity constraint**, sometimes referred to as a **trivial constraint**, of the form $x_j \geqslant 0$ for each variable x_j in the model.

The steps involved in formulating a linear programming model are summarized in the procedure below, which (as with the mathematical modelling process) is intended not as a rigorous set of rules but as a rough guide.

Procedure 1.1 Formulating linear programming models

Given a precise statement of a linear programming problem, including a specified purpose and any appropriate data and assumptions, the problem may be formulated as a linear programming model as follows.

(a) Identify the *objective* of the model (for example maximizing profit or minimizing cost) and decide on the *units* in which the *objective function* is to be measured (for example units of currency).

(b) Identify the *variables* and decide on the *units* in which each is to be measured.

(c) Identify the *constraints* and *parameters* for the problem.

(d) Assign algebraic symbols to the objective function (usually z) and to the variables (usually $x_1, x_2, \ldots, x_n$), and write down precise definitions, including units of measurement, for all of these. If necessary, assign algebraic symbols to the parameters as well.

(e) Using a table of parameter values or otherwise, identify the linear relationships between the objective function and the variables and between the constraints and the variables, being careful to use consistent units when identifying these relationships.

(f) Write down the objective of the problem in the form

$$\text{optimize } z = c_1x_1 + c_2x_2 + \cdots + c_nx_n.$$

The term 'optimize' here should be replaced, in any particular problem, by 'maximize' or 'minimize' as appropriate.

(g) Write down the **non-trivial constraints** where the ith non-trivial constraint is a linear relationship of one of the following forms:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leqslant b_i;$$
$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geqslant b_i;$$
$$a_{i1}x_i + a_{i2}x_2 + \cdots + a_{in}x_n = b_i.$$

(h) Write down the non-negativity or trivial constraints

$$x_j \geqslant 0 \qquad (j = 1, 2, \ldots, n).$$

When deciding on the units of measurement for the objective function and the variables, consideration needs to be given to the *scaling* of the model. It is desirable to avoid very large or very small coefficients in the constraints. This is partly because the simplex method, which we shall use to solve such problems, involves solving systems of linear equations based on the linear constraints, and we have seen in *Unit I.3* how badly scaled systems can lead to poor numerical solutions. Also, simple scaling of the objective function and constraints can avoid the need to work with very large or very small numbers, with lots of zeros before or after the decimal point. Thus, for example, it may be more sensible to work in thousands of pounds or even fractions of a penny rather than pounds. Remember, however, to scale the values of the parameters to match any scaling of the variables and objective function: both the constraints and the objective function need to be dimensionally consistent, in the terminology of *Unit I.5*.

The *parameters* in the general formulation of a linear programming model in Procedure 1.1 are given algebraic symbols: the coefficients c_j of the objective function, the coefficients a_{ij} of the constraints and the right-hand sides b_i of the constraints. This general formulation, with symbols for the parameters, is in keeping with the general form of model recommended in *Unit I.5*. In a sense, when modelling a specific problem, what we are doing is assigning numerical values to these symbols. However, as we remarked earlier, when modelling a particular linear programming problem we would rarely write down the general form but rather go directly to the specific form, with numerical values for the parameters (as was done in Example 1.1), while keeping the general form in mind as a blueprint.

Note that, although all the variables identified need to play a role somewhere in the model, they need not all appear in the objective function (i.e. some of the coefficients c_j in z may be zero).

You may have noticed from Procedure 1.1 that non-trivial constraints can include $\leqslant$, $\geqslant$ or $=$ signs, and not just $\leqslant$ signs as in Example 1.1. Any particular model may include non-trivial constraints of more than one type, i.e. it may include **mixed constraints**. The following exercise provides an example of a model with mixed constraints.

Exercise 1.1

A farmer has ten pigs and feeds them on fish meal and sterilized meat scraps. He wants to feed the pigs at minimum cost but must ensure that they get enough protein, enough of certain specified amino acids and not too much calcium. Fish meal costs £150 a tonne while meat scraps cost £120 a tonne. Each kilogram of fish meal contains 600 gm of protein, 50 gm of calcium and 180 gm of the specified amino acids. Each kilogram of meat scraps contains 500 gm of protein, 110 gm of calcium and 50 gm of the amino acids. Each pig requires at least 1.6 kg of protein and 0.3 kg of the amino acids a day, but must not have more than 0.3 kg of calcium a day. Formulate the problem as a linear programming model.

1 tonne = 1000 kilograms.

Example 1.1 and Exercise 1.1 showed (in simplified form) two typical applications of linear programming: *manufacturing problems*, where a maximum profit is required subject to limited resources (in terms of staffing or machinery), and *diet problems*, where a minimum cost diet is required subject to restrictions on nutrients. We now look at an example of a third typical application: *blending problems*, where the maximum profit is required on a product subject to quality constraints as well as limited resources.

Example 1.2

A food manufacturer buys edible oils in their raw state, and refines and blends them to produce margarine. The raw oils can be vegetable or non-vegetable. The vegetable oils are ground-nut, soya-bean and palm. The non-vegetable oils are lard and fish. The margarine sells at £2400 a tonne, and the manufacturer can sell as much margarine as can be made. The prices of the raw oils are shown in Table 1.2. Sufficient quantities of all five oils can be assumed to be readily available.

Table 1.2

oil	ground-nut	soya-bean	palm	lard	fish
cost (£ per tonne)	960	1600	1600	2200	1900
hardness	1.2	3.4	8.0	10.8	8.3

We assume that 'hardness' is measured in appropriate units.

To avoid contamination, vegetable and non-vegetable oils are refined separately, using different equipment. It is possible to refine up to 1000 tonnes of vegetable oil and up to 800 tonnes of non-vegetable oil in a month. When refined, some or all of the oils are blended to produce the margarine.

The quantities of oil lost during the refining and blending processes are negligible. The hardnesses of the refined oils, given in Table 1.2, are assumed to combine linearly to give the hardness of the margarine. To ensure that the margarine spreads easily but is not runny, the hardness of the margarine must be between 5.6 and 7.4, measured in the same units as the hardnesses of the oils in Table 1.2. All running costs can be taken to be fixed. The manufacturer wants to know how much of each type of oil to use in order to maximize profit while maintaining the quality of the margarine.

The assumed linearity of the relationship means, for example, that if x_1 tonnes of oil A with hardness h_1 is combined with x_2 tonnes of oil B with hardness h_2 to give $(x_1 + x_2)$ tonnes of margarine with hardness h then $x_1h_1 + x_2h_2 = (x_1 + x_2)h$.

Formulation of Example 1.2

We follow the steps of Procedure 1.1.

(a) The purpose of the model is to maximize profit, subject to constraints on resources, while maintaining quality. Since the refining capacities are given per month, it is sensible to work on a monthly basis. We could therefore take maximizing monthly profit as the objective. However, since the running costs are fixed, maximizing monthly profit is equivalent to maximizing monthly income. Therefore, as omitting the fixed running cost will simplify the model, we shall take the objective as maximizing monthly income.

This monthly income could be measured in pounds. However, looking at the quantities involved, it would seem more sensible to work in units of £10 000.

(b) The variables are the quantities of each oil used per month in blending the margarine. Also, as it will be useful to know the total quantity of margarine manufactured per month, we shall regard this as a variable too.

If we did not introduce a variable for the total quantity of margarine, we would have to determine it by adding the quantities of the refined oils.

Exercise 1.2

Choose convenient units for the variables (part of step (b)).

Exercise 1.3

Describe briefly in words the constraints for the problem, and identify the parameters (step (c)).

(*Hint*: The introduction of a variable for the total quantity of margarine manufactured per month results in an extra constraint, over and above those mentioned in the specification of the problem.)

(d) We now assign algebraic symbols to the objective function and the variables, giving precise definitions, including units of measurement:

z the monthly profit, in tens of thousands of pounds;
x_1 the quantity, in hundreds of tonnes, of ground-nut oil used per month;
x_2 the quantity, in hundreds of tonnes, of soya-bean oil used per month;
x_3 the quantity, in hundreds of tonnes, of palm oil used per month;
x_4 the quantity, in hundreds of tonnes, of lard used per month;
x_5 the quantity, in hundreds of tonnes, of fish oil used per month;
x_6 the quantity, in hundreds of tonnes, of margarine manufactured per month.

(e) We next identify the linear relationships for the objective function and the constraints. We can do this by extending Table 1.2, and making suitable adjustments to the numerical values to take into account the units of measurement decided on in (a) and (b), and listed in (d). This results in Table 1.3.

Table 1.3

	ground-nut oil	soya-bean oil	palm oil	lard	fish oil	upper limit	lower limit
cost (£10 000s per 100 tonnes)	9.6	16	16	22	19	—	—
refining of vegetable oils (100s of tonnes)	✓	✓	✓	—	—	10	—
refining of non-vegetable oils (100s of tonnes)	—	—	—	✓	✓	8	—
hardness (hardness units)	1.2	3.4	8.0	10.8	8.3	7.4	5.6

The ticks in rows 2 and 3 indicate the vegetable and non-vegetable oils respectively.

The quantity constraint does not appear in Table 1.3, but we can deduce the relevant linear relationship directly from its description (in the solution to Exercise 1.3). The table also omits the income from selling the margarine, which is 24, in units of £10 000s per 100 tonnes.

Exercise 1.4

Determine the objective function and objective of the problem in terms of $x_1, \ldots, x_6$ (step (f)).

(g) We now want to express the constraints algebraically in terms of the variables.

Exercise 1.5

Express the refining capacity constraints in terms of the variables.

The hardness constraints require some thought. Remember that the hardnesses of the components combine linearly to give the hardness of the margarine, which must be at least 5.6. Using Table 1.3 this means that we must have

$$1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 \geqslant 5.6x_6.$$

If, for consistency, we keep all variables to the left-hand side of inequality or equality signs, this becomes

$$1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 5.6x_6 \geqslant 0.$$

Exercise 1.6

Write down a similar inequality for the upper hardness constraint.

We must also remember the quantity constraint, which can be written as

$$x_1 + x_2 + x_3 + x_4 + x_5 = x_6$$

or, in the format of the other constraints, as

$$x_1 + x_2 + x_3 + x_4 + x_5 - x_6 = 0.$$

(h) Finally, as the manufacturer cannot use negative quantities of oil, we have the trivial constraints

$$x_1, x_2, x_3, x_4, x_5, x_6 \geqslant 0.$$

The linear programming model for Example 1.2 is thus:

maximize $z = -9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5 + 24x_6$

subject to

$$\begin{aligned}
x_1 + x_2 + x_3 &\leqslant 10 && \text{(vegetable refining capacity)}\\
x_4 + x_5 &\leqslant 8 && \text{(non-vegetable refining capacity)}\\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 5.6x_6 &\geqslant 0 && \text{(lower hardness)}\\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 7.4x_6 &\leqslant 0 && \text{(upper hardness)}\\
x_1 + x_2 + x_3 + x_4 + x_5 - x_6 &= 0 && \text{(quantity)}
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geqslant 0 \quad \blacksquare$$

Exercise 1.7

A smallholder rears a number of bullocks, pigs and free-range hens. The land requirements per animal and the food costs, labour requirements and income per animal per year are shown in Table 1.4.

Table 1.4

	land requirements (hectares)	food costs (£ per year)	labour requirements (hours per year)	income (£ per year)
each hen	0.01	20	—	30
each pig	0.2	120	20	240
each bullock	1	240	30	500

The smallholder has 25 hectares of land available for animal rearing and £4000 to spend on food per year. His children, who feed the hens, can manage no more than 200 hens. He feeds the bullocks and pigs himself, and has up to 1000 hours per year available for this. He wants to know how many of each animal to keep in order to maximize his income. Formulate the problem as a linear programming model.

We have now formulated four linear programming models, all of the general form described on the following page.

Definition

The **general form of a linear programming model** consists of:

(a) an **objective** to maximize or minimize some quantity z, known as the **objective function**, which is expressed as a linear combination of the **variables** $x_1, x_2, \ldots, x_n$;

(b) **non-trivial constraints** on the variables, which are inequalities or equations involving linear combinations of the variables;

(c) **trivial** or **non-negativity constraints** on the variables to ensure that each one is positive or zero.

It can be expressed mathematically as follows.

$$\begin{aligned}
&\text{optimize } z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
&\text{subject to} \\
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leqslant / = / \geqslant b_1 \\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leqslant / = / \geqslant b_2 \\
&\vdots \\
&a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leqslant / = / \geqslant b_m \\
&\qquad x_1, x_2, \ldots, x_n \geqslant 0
\end{aligned}$$

The term 'optimize' here should be replaced, in any particular problem, by 'maximize' or 'minimize' as appropriate.

The c_j $(j = 1, \ldots, n)$, a_{ij} $(i = 1, \ldots, m; j = 1, \ldots, n)$ and b_i $(i = 1, \ldots, m)$ are the **parameters** of the model.

In the examples we have considered so far, we have assumed that running costs are constant and so have omitted these in our formulation of the objective function. Strictly speaking we should include such constant terms in the objective function, though it is often convenient to omit them (as we have done). In so doing, however, we are effectively changing the objective function from the actual profit (or cost), z' say, to

$$z = \sum_{j=1}^{n} c_j x_j = z' - c$$

where c represents the total of the constant terms. It is important, therefore, when solving linear programming problems, to remember to add the value of any constant terms omitted from the objective function to the maximum (or minimum) value of z in order to determine the correct value for the maximum profit (or minimum cost).

1.2 Standard form

Our next aim is to express the general form of a linear programming model in terms of matrices. Doing so will enable us to obtain some powerful theoretical results (in *Unit II.2*).

If we write $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and $\mathbf{c} = [c_1, c_2, \ldots, c_n]^T$, we can express the objective function, z, as

$$z = \mathbf{c}^T\mathbf{x}.$$

In problems such as Examples 1.1 and 1.2 and Exercise 1.7, where we are maximizing income, the parameter c_j often represents the contribution to income made by the selling price of product x_j, so $\mathbf{c}$ is sometimes called the **price vector**. In minimization problems, such as Exercise 1.1, the parameters c_j often represent costs, so $\mathbf{c}$ is instead known as the **cost vector**. In fact, as any maximization problem

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$

can be regarded as a minimization problem by rewriting it in the form

$$\text{minimize } (-z) = -\mathbf{c}^T\mathbf{x},$$

we often regard the components of a price vector as **negative costs** and use the term *cost vector* for both maximization and minimization problems.

Before developing the rest of the matrix formulation of the general form of a linear programming model, it may help to look at an example. The model for the table manufacturing problem (Example 1.1) is:

$$\begin{aligned}
&\text{maximize } z = 120x_1 + 150x_2 \\
&\text{subject to} \\
&2x_1 + 3x_2 \leqslant 30 \qquad \text{(joinery)} \\
&\tfrac{2}{3}x_1 + 2x_2 \leqslant 16 \qquad \text{(prefinishing)} \\
&5\tfrac{1}{3}x_1 + 4x_2 \leqslant 64 \qquad \text{(final finishing)} \\
&x_1, x_2 \geqslant 0
\end{aligned}$$

If $\mathbf{c} = [120, 150]^T$ and $\mathbf{x} = [x_1, x_2]^T$ then the objective becomes

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}.$$

If the coefficients on the left-hand sides of the non-trivial constraints are denoted by the matrix $\mathbf{A}$ and the right-hand sides by $\mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ \frac{2}{3} & 2 \\ 5\frac{1}{3} & 4 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 30 \\ 16 \\ 64 \end{bmatrix},$$

then we would like to express the non-trivial constraints as a relationship between the vector $\mathbf{Ax}$, representing the left-hand sides, and the vector $\mathbf{b}$, representing the right-hand sides. The obvious way to write this is

$$\mathbf{Ax} \leqslant \mathbf{b},$$

but we must be sure of what we mean by an inequality sign between two vectors. This is defined, in the context of constrained optimization, in the same way as equality between two vectors, i.e. it must apply to *every element*.

Definition

For n-dimensional vectors $\mathbf{x} = [x_1, \ldots, x_n]^T$ and $\mathbf{y} = [y_1, \ldots, y_n]^T$, the inequalities $\leqslant$ and $\geqslant$ are defined as follows:

(a) $\mathbf{x} \leqslant \mathbf{y}$ if and only if $x_i \leqslant y_i$, $i = 1, \ldots, n$;

(b) $\mathbf{x} \geqslant \mathbf{y}$ if and only if $x_i \geqslant y_i$, $i = 1, \ldots, n$.

The strict inequalities $<$ and $>$ can be defined for vectors in a similar way, though strict inequalities are not used in linear or integer programming.

In light of this definition, $\mathbf{Ax} \leqslant \mathbf{b}$ does indeed have the desired meaning.

Exercise 1.8

Express the non-negativity constraints in vector notation.

We can now write the general form of the linear programming model for the table manufacturing problem in matrix notation as

$$\begin{aligned}&\text{maximize } z = \mathbf{c}^T\mathbf{x}\\&\text{subject to } \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}\end{aligned}$$

where $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{x}$ are defined as above.

Although we have succeeded in expressing the general form of a particular linear programming model in matrix notation, we cannot easily do so for any linear programming model. Consider, for example, the model for the pig feeding problem (Exercise 1.1):

$$\begin{aligned}&\text{minimize } z = 15x_1 + 12x_2\\&\text{subject to}\\&\begin{aligned} 0.6x_1 + \ 0.5x_2 &\geqslant 16 \quad && \text{(protein)}\\ 0.05x_1 + 0.11x_2 &\leqslant \ 3 \quad && \text{(calcium)}\\ 0.18x_1 + 0.05x_2 &\geqslant \ 3 \quad && \text{(amino acids)}\\ x_1, x_2 &\geqslant 0\end{aligned}\end{aligned}$$

If we let $\mathbf{c} = [15, 12]^T, \mathbf{x} = [x_1, x_2]^T$,

$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.5 \\ 0.05 & 0.11 \\ 0.18 & 0.05 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 16 \\ 3 \\ 3 \end{bmatrix},$$

then we can write $z = \mathbf{c}^T\mathbf{x}$ and $\mathbf{x} \geqslant \mathbf{0}$, but we have a difficulty with the relationship between $\mathbf{A}\mathbf{x}$ and $\mathbf{b}$ because the constraints are *mixed*. To overcome this difficulty we need to rewrite the model in a more manageable *standard form*.

Definition

A linear programming model is in **standard form** if it is expressed in the form:

$$\begin{aligned}&\text{maximize } z = \mathbf{c}^T\mathbf{x}\\&\text{subject to } \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}\end{aligned}$$

Somewhat confusingly, the standard form is referred to as the *canonical form* in some texts (see Subsection 1.3).

To convert the pig feeding model to standard form, we first need to change the objective to a maximization:

$$\text{maximize } z' = -15x_1 - 12x_2,$$

where $z' = -z$.

The protein and amino acids constraints then need to be expressed as $\leqslant$ constraints, which we can do by changing all the signs, to give:

$$\begin{aligned} -0.6x_1 - \ 0.5x_2 &\leqslant -16 \quad && \text{(protein)}\\ -0.18x_1 - 0.05x_2 &\leqslant \ -3 \quad && \text{(amino acids)}\end{aligned}$$

Thus the standard form of the pig feeding model is

$$\begin{aligned}&\text{maximize } z' = \mathbf{c}^T\mathbf{x}\\&\text{subject to } \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}\end{aligned}$$

where $\mathbf{c} = [-15, -12]^T, \mathbf{x} = [x_1, x_2]^T$,

$$\mathbf{A} = \begin{bmatrix} -0.6 & -0.5 \\ 0.05 & 0.11 \\ -0.18 & -0.05 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -16 \\ 3 \\ -3 \end{bmatrix}.$$

We have seen how minimizations can be changed to maximizations and $\geqslant$ constraints to $\leqslant$ ones simply by changing signs. However, some models (such as the model for the margarine blending problem of Example 1.2) also include equality constraints. These too must be eliminated if the model is to be expressed in standard form. In Example 1.2 the quantity constraint was

$$x_1 + x_2 + x_3 + x_4 + x_5 - x_6 = 0.$$

We can treat this in one of two ways: we can *either* eliminate the constraint, by replacing x_6 by $(x_1 + x_2 + x_3 + x_4 + x_5)$ wherever it occurs, *or* replace the constraint by the following two inequality constraints.

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 + x_5 - x_6 &\leqslant 0 \\ -x_1 - x_2 - x_3 - x_4 - x_5 + x_6 &\leqslant 0 \end{aligned}$$

Which method we choose will depend on the problem we are modelling. The latter method involves more variables and constraints, and so will require more computation. So, unless there is a good reason for preferring it, such as requiring a value for the additional variable to be output on solving the model or simplifying the formulation of the problem, then the former method should be used.

As we deliberately introduced the variable x_6 when modelling the margarine blending problem, we might well choose the latter method in this case.

Exercise 1.9

The model for the margarine blending problem (Example 1.2) is:

maximize $z = -9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5 + 24x_6$

subject to

$$\begin{aligned}
x_1 + x_2 + x_3 &\leqslant 10 && \text{(vegetable refining capacity)} \\
x_4 + x_5 &\leqslant 8 && \text{(non-vegetable refining capacity)} \\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 5.6x_6 &\geqslant 0 && \text{(lower hardness)} \\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 7.4x_6 &\leqslant 0 && \text{(upper hardness)} \\
x_1 + x_2 + x_3 + x_4 + x_5 - x_6 &= 0 && \text{(quantity)}
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geqslant 0$$

Express this model in standard form by:

(a) replacing the variable x_6 by $x_1 + x_2 + x_3 + x_4 + x_5$;

(b) replacing the quantity constraint by two inequalities.

In all the problems we have met so far, the non-negativity constraints arose naturally from the situation under consideration. This will not always be the case. For example, suppose the variable x_j represents the amount of money held in a current account (in some banking optimization problem). The account may be overdrawn, so x_j could take negative values. To convert a model involving x_j to standard form, we can replace it wherever it occurs by $x_j^+ - x_j^-$, where x_j^+ and x_j^- are both non-negative. This allows the original variable x_j to take negative values, when $x_j^- > x_j^+$, but ensures that all the variables in the model take only non-negative values. A variable that can take both positive and negative values is called a **free variable**. In general, a free variable should be replaced by two non-negative variables when converting a model to standard form.

Procedure 1.2 Expressing a linear programming model in standard form

To express a linear programming model in standard form as

maximize $z = \mathbf{c}^T\mathbf{x}$
subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

proceed as follows.

(a) Remove any constant term c from the objective function by replacing $z' = \mathbf{c}^T\mathbf{x} + c$ by $z = \mathbf{c}^T\mathbf{x}$, where $z = z' - c$.

(b) If the problem is posed as a minimization, modelled as

minimize $z' = (\mathbf{c}')^T\mathbf{x}$,

rewrite it as

maximize $z = \mathbf{c}^T\mathbf{x}$,

where $z = -z'$ and $\mathbf{c} = -\mathbf{c}'$.

(c) Remove any free variables x_j by setting $x_j = x_j^+ - x_j^-$, where $x_j^+ \geqslant 0$, $x_j^- \geqslant 0$, and replacing x_j by $x_j^+ - x_j^-$ wherever it occurs in the objective function and the constraints.

More usually we use x_j and x_q, say, rather than x_j^+ and x_j^-, where q is a subscript not already in use in the model. We thus replace x_j in the model by $x_j - x_q$, where now $x_j \geqslant 0$ (and $x_q \geqslant 0$).

(d) Remove any equality constraint, such as

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i,$$

either by eliminating one of the variables, x_j say, by writing

$$x_j = \frac{1}{a_{ij}}\left(b_i - \sum_{k \neq j} a_{ik}x_k\right) \qquad (a_{ij} \neq 0)$$

and replacing x_j, wherever it occurs in the objective function and the constraints, by this combination of the other $n-1$ variables *or* by replacing it by the following two $\leqslant$ inequalities:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leqslant b_i$$
$$-a_{i1}x_1 - a_{i2}x_2 - \cdots - a_{in}x_n \leqslant -b_i.$$

Remember that this second option should only be chosen if there is a good reason for doing so. For example, eliminating x_j means losing the constraint $x_j \geqslant 0$, which may cause problems.

(e) Change each $\geqslant$ constraint, such as

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geqslant b_i,$$

to a $\leqslant$ constraint by changing all the signs, to give

$$-a_{i1}x_1 - a_{i2}x_2 - \cdots - a_{in}x_n \leqslant -b_i.$$

(f) If necessary and if desired, renumber the variables.

Exercise 1.10

Express the following linear programming model in standard form.

minimize $x_1 + 2x_2 + 6$
subject to

$$\begin{aligned} x_1 - x_2 - x_3 &\leqslant -7 \\ 2x_1 - x_4 &= 8 \\ x_2 - x_3 + 2x_4 &\geqslant 0 \\ x_1, x_3, x_4 &\geqslant 0 \end{aligned}$$

1.3 Canonical form

The standard form of the previous subsection is useful in that it allows some powerful theoretical results to be applied to linear programming models, as you will see in the next unit. That form of the model does not, however, lead to a general method for solving linear programming models, as we have no means of solving the inequality $\mathbf{Ax} \leqslant \mathbf{b}$. The general method of solution that we consider in this course, the simplex method, requires us to express the inequality constraints as equalities, giving $\mathbf{Ax} = \mathbf{b}$ instead. This can be done by introducing additional variables.

The simplex method is discussed in Sections 3 and 4 of this unit, and in *Unit II.2*.

Definition

A linear programming model is in **canonical form** if it is expressed in the form

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$$

where the signs are chosen so that $\mathbf{b} \geqslant \mathbf{0}$.

Somewhat confusingly, the canonical form is referred to as the *standard form* in some texts.

Example 1.3

The standard form of the matrix formulation of the table manufacturing problem (Example 1.1) was found, in Subsection 1.2, to be

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{Ax} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$$

where $\mathbf{c} = [120, 150]^T, \mathbf{x} = [x_1, x_2]^T$,

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ \frac{2}{3} & 2 \\ 5\frac{1}{3} & 4 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 30 \\ 16 \\ 64 \end{bmatrix}.$$

Express this in canonical form.

Solution

The objective is of the required form and $\mathbf{b} \geqslant \mathbf{0}$, but the constraints $\mathbf{Ax} \leqslant \mathbf{b}$ need to be converted to equalities. To do this we introduce new non-negative variables into the model. Consider the joinery constraint

$$2x_1 + 3x_2 \leqslant 30.$$

This can be expressed as

$$2x_1 + 3x_2 + x_3 = 30$$

where $x_3 \geqslant 0$. The new variable x_3 is called a **slack variable** because it represents the 'slack' in the joinery capacity: if there is no slack, $x_3 = 0$ and the full joinery capacity of 30 person-hours is being used; otherwise x_3 represents the number of spare person-hours of joinery capacity. Similarly, slack variables x_4 and x_5 can be introduced into the prefinishing and final finishing constraints to give

$$\tfrac{2}{3}x_1 + 2x_2 + x_4 = 16$$
$$5\tfrac{1}{3}x_1 + 4x_2 + x_5 = 64$$

where $x_4, x_5 \geqslant 0$. This time, x_4 and x_5 represent the number of spare person-hours of prefinishing and final finishing capacity respectively.

Thus, in terms of matrices, the canonical form is

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$$

where $\mathbf{c} = [120, 150, 0, 0, 0]^T, \mathbf{x} = [x_1, x_2, x_3, x_4, x_5]^T$,

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 1 & 0 & 0 \\ \frac{2}{3} & 2 & 0 & 1 & 0 \\ 5\frac{1}{3} & 4 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 30 \\ 16 \\ 64 \end{bmatrix}. \quad \blacksquare$$

Notice that the $\mathbf{A}$, $\mathbf{c}$ and $\mathbf{x}$ in the canonical form of the model are not the same as in the standard form.

When converting a linear programming model to canonical form, we start from whichever form of the model we have available or is most convenient.

Example 1.4

The general form of the model for the pig feeding problem (Exercise 1.1) is:

$$\text{minimize } z = 15x_1 + 12x_2$$
$$\text{subject to}$$
$$0.6x_1 + 0.5x_2 \geqslant 16$$
$$0.05x_1 + 0.11x_2 \leqslant 3$$
$$0.18x_1 + 0.05x_2 \geqslant 3$$
$$x_1, x_2 \geqslant 0$$

Express this model in canonical form.

Solution

We must first change the objective to a maximization:

$$\text{maximize } z' = -15x_1 - 12x_2.$$

We have $\mathbf{b} \geqslant \mathbf{0}$ but the non-trivial constraints are not equalities. The protein constraint, for example, is

$$0.6x_1 + 0.5x_2 \geqslant 16.$$

We require that the new variable which we introduce should be non-negative, so we must *subtract* a variable to turn this constraint into an equality:

$$0.6x_1 + 0.5x_2 - x_3 = 16, \qquad x_3 \geqslant 0.$$

x_3 is called a **surplus variable** because it represents the amount of protein in the feed that is *surplus* to the bare minimum requirements. The calcium constraint can be converted to an equality using a slack variable, x_4:

$$0.05x_1 + 0.11x_2 + x_4 = 3, \qquad x_4 \geqslant 0.$$

The amino acids constraint requires another surplus variable, x_5:

$$0.18x_1 + 0.05x_2 - x_5 = 3, \qquad x_5 \geqslant 0.$$

Thus, in terms of matrices, the canonical form is

$$\text{maximize } z' = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$$

where $\mathbf{c} = [-15, -12, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]^T$,

$$\mathbf{A} = \begin{bmatrix} 0.6 & 0.5 & -1 & 0 & 0 \\ 0.05 & 0.11 & 0 & 1 & 0 \\ 0.18 & 0.05 & 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 16 \\ 3 \\ 3 \end{bmatrix}. \quad \blacksquare$$

Procedure 1.3 Expressing a linear programming model in canonical form

To express a linear programming model in canonical form as

maximize $z = \mathbf{c}^T\mathbf{x}$
subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{b} \geqslant \mathbf{0}$, proceed as follows.

(a) Remove any constant term c from the objective function by replacing $z' = \mathbf{c}^T\mathbf{x} + c$ by $z = \mathbf{c}^T\mathbf{x}$, where $z = z' - c$.

(b) If the problem is posed as a minimization, modelled as

minimize $z' = \mathbf{c}'^T\mathbf{x}$,

rewrite it as

maximize $z = \mathbf{c}^T\mathbf{x}$.

where $z = -z'$ and $\mathbf{c} = -\mathbf{c}'$.

(c) Remove any free variables x_j by setting $x_j = x_j^+ - x_j^-$, where $x_j^+ \geqslant 0$, $x_j^- \geqslant 0$, and replacing x_j by $x_j^+ - x_j^-$ wherever it occurs in the objective function and the constraints.

More usually we use x_j and x_q, say, rather than x_j^+ and x_j^-, where q is a subscript not already in use in the model. We thus replace x_j in the model by $x_j - x_q$, where now $x_j \geqslant 0$ (and $x_q \geqslant 0$).

(d) For any $b_i < 0$, change all the signs in the corresponding constraint and reverse the direction of any inequality sign.

(e) (i) Convert any constraint of the form

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leqslant b_i$$

to an equality by adding a *slack variable*, x_{n+i}, to give:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + x_{n+i} = b_i, \qquad x_{n+i} \geqslant 0.$$

(ii) Convert any constraint of the form

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geqslant b_i$$

to an equality by subtracting a *surplus variable*, x_{n+i}, to give

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - x_{n+i} = b_i, \qquad x_{n+i} \geqslant 0.$$

Note that if, after removing any free variables, a linear programming model has n variables, m non-trivial inequality constraints and k equality constraints, then its canonical form will have $n + m$ variables and $m + k$ equality constraints.

(f) Extend $\mathbf{c}$ to include zero elements c_{n+i} for the slack and surplus variables x_{n+i}. If necessary and if desired, renumber the variables.

Note that the variables that are not slack or surplus are sometimes referred to as **main variables**. Also, for simplicity, we shall often refer collectively to the slack and surplus variables of a model as **slack variables**.

Exercise 1.11

The model of Exercise 1.10 is as follows.

$$\begin{aligned}
&\text{minimize } x_1 + 2x_2 + 6 \\
&\text{subject to} \\
&x_1 - x_2 - x_3 \leqslant -7 \\
&2x_1 - x_4 = 8 \\
&x_2 - x_3 + 2x_4 \geqslant 0 \\
&x_1, x_3, x_4 \geqslant 0
\end{aligned}$$

Express this model in canonical form, using matrix notation.

Exercise 1.12

The general form of the model for the margarine blending problem (Example 1.2) is:

maximize $z = -9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5 + 24x_6$

subject to

$$\begin{aligned}
x_1 + x_2 + x_3 &\leqslant 10 \\
x_4 + x_5 &\leqslant 8 \\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 5.6x_6 &\geqslant 0 \\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 7.4x_6 &\leqslant 0 \\
x_1 + x_2 + x_3 + x_4 + x_5 - x_6 &= 0 \\
x_1, x_2, x_3, x_4, x_5, x_6 &\geqslant 0.
\end{aligned}$$

Express this model in canonical form, using matrix notation.

Exercise 1.13

The general (and standard) form of the model for the smallholder problem (Exercise 1.7) is:

maximize $z = 30x_1 + 240x_2 + 500x_3$

subject to

$$\begin{aligned}
0.01x_1 + 0.2x_2 + x_3 &\leqslant 25 \\
20x_1 + 120x_2 + 240x_3 &\leqslant 4000 \\
20x_2 + 30x_3 &\leqslant 1000 \\
x_1 &\leqslant 200 \\
x_1, x_2, x_3 &\geqslant 0.
\end{aligned}$$

Express this model in canonical form, using matrix notation.

End-of-section Exercises

Exercise 1.14

Formulate the following linear programming problem as a model in both standard and canonical form, using matrix notation.

The Christmas Hamper Company has a store containing 10 000 kg of nuts, 4000 Christmas puddings, 2000 bottles of sherry and 1500 turkeys. It intends to use these goods to make up three different sorts of hampers with contents as in Table 1.5. The company has set the price of Hamper A at £50, Hamper B at £30 and Hamper C £25, and wants to know how many of each sort of hamper it should aim to sell in order to maximize revenue.

Table 1.5

hamper	A	B	C
nuts (kg)	6	4	8
puddings	3	2	0
sherry (bottles)	2	0	3
turkeys	1	1	0

Exercise 1.15

Formulate the following linear programming problem as a model in both standard and canonical form, using matrix notation.

Haldon Gardens grows three types of plant by taking cuttings which then need either potting off, potting on or both. Azaleas require potting on, rhododendrons require potting off and potting on, while camellias require potting off. There is only capacity to take 600 cuttings, to pot off 400 plants and to pot on 300 plants a day. Azaleas sell for £3 each, rhododendrons for £5 and camellias for £4. The demand for rhododendrons and camellias is unlimited, but not more than 80 azaleas a day can be sold. The head gardener wants to know how many of each type of plant to treat per day in order to maximize daily takings. (Assume that there is a steady flow of plants through the process, so that the number of plants sold each day is the same as the number treated. Also, assume that treating a given number of plants means taking that number of cuttings but also potting on/off the same number of plants.)

Potting off is the act of potting seedlings in their their own pot for the first time, while potting on is the act of repotting a plant in a larger pot to give it more room to grow.

Exercise 1.16

Formulate the following linear programming problem as a model in general, standard and canonical forms, using matrix notation in the last two cases.

Hairways, a manufacturer of hair-related products, is planning to bring out a new shampoo. It wants to include two fashionable ingredients, Protein-Plus and Sheen, and its research staff has deemed that these ingredients need to comprise at least 10% and 30% of the capacity of the shampoo, respectively, if they are to be effective. It also wants to include a conditioner in the product, but recent legislation does not permit a shampoo to contain more than 25% conditioner. Hairways' usual supplier of chemical products has two suitable products available that contain Protein-Plus, Sheen and conditioner:

- Special H, which contains 20% Protein-Plus, 20% Sheen and 20% conditioner, and costs £1.50 for ten litres;
- Super Triple-X, which contains 5% Protein-Plus, 50% Sheen and 30% conditioner, and costs £2.25 for ten litres.

Hairways wants to use a combination of these products, plus, if needed, a water-based product of negligible cost to make up the volume, in manufacturing its new shampoo. The water-based product does not contain any of Protein-Plus, Sheen or conditioner. All costs associated with manufacturing the shampoo can be taken to be fixed. The amount of wastage in the manufacturing process is negligible. Hairways wants to know how much of each of Special H and Super Triple-X to purchase in order to manufacture 1000 litres of the new shampoo at minimum cost.

2 Graphical interpretation

In this section, we shall look at two-dimensional linear programming models and interpret them graphically. In Subsection 2.1 we shall investigate the different sorts of region that can be defined by the constraints. In Subsection 2.2, we shall see how such regions may be used to solve the models graphically. We shall then go on, in Subsection 2.3, to examine the different types of solution that are possible. Finally, in Subsection 2.4, we shall look briefly at a graphical representation of the canonical form of a model.

We shall also, briefly, mention how the ideas extend to more than two dimensions. A number of new terms will be defined, many of which will prove useful later. We shall, in Subsections 2.1–2.3, generally assume that the model is in standard form, though we shall sometimes consider minimization problems and $\geqslant$ constraints. We shall not consider equality constraints, except in Subsection 2.4.

2.1 Feasible regions

If a linear programming model involves only two variables, x_1 and x_2 say, then pairs of values for the variables can be represented as points in the (x_1, x_2)-plane. A linear constraint can then restrict the points (x_1, x_2) to lie in the half-plane on one side of a line. For example, the constraint

$$2x_1 + 3x_2 \leqslant 30$$

restricts (x_1, x_2) to lie in the unshaded area below the line

$$2x_1 + 3x_2 = 30,$$

as shown in Figure 2.1.

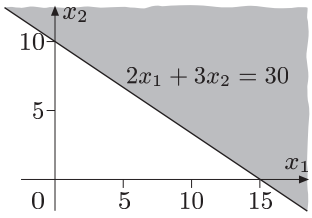


Figure 2.1

To show the region of possible values of (x_1, x_2) when several constraints must all be satisfied at the same time, we must take the *intersection* of the regions for each constraint. For example, if we take the non-negativity constraints $x_1 \geqslant 0$ (which restricts us to the half-plane to the right of the x_2-axis) and $x_2 \geqslant 0$ (which restricts us to the half-plane above the x_1-axis), we obtain the unshaded region shown in Figure 2.2(a). If we then add the constraint $2x_1 + 3x_2 \leqslant 30$, we obtain the triangular region shown in Figure 2.2(b).

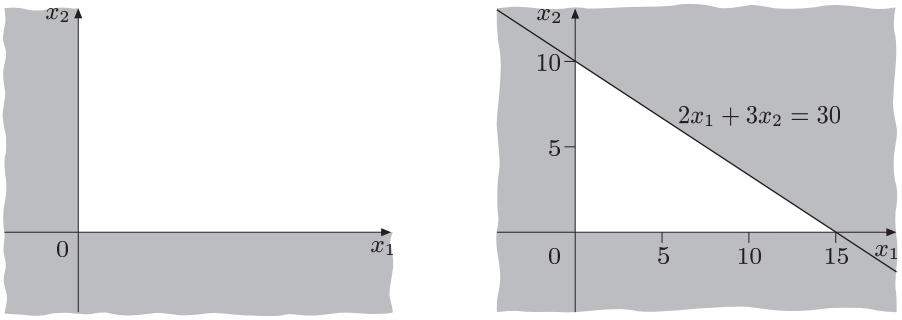


Figure 2.2 (a) $x_1 \geqslant 0$ and $x_2 \geqslant 0$ (b) $x_1 \geqslant 0$, $x_2 \geqslant 0$ and $2x_1 + 3x_2 \leqslant 30$

The region containing the points that satisfy all the constraints of a linear programming model is known as the **feasible region** for the model. The points in the feasible region are known as **feasible points**. The lines forming the boundary of a feasible region are known as **constraint bounds**.

The points on the boundary of the feasible region are known as **boundary points**. Note that, because inequality constraints in linear programming models are never strict, the feasible region always includes its boundary points, i.e. all boundary points are feasible points.

We shall show the feasible region of a two-dimensional linear programming problem as the *unshaded* part of the (x_1, x_2)-plane, as in Figures 2.1 and 2.2. Note that if, having drawn the line for a constraint, you are unsure which side of it should be shaded, then you can find out by testing whether a convenient point (such as the origin, $(0, 0)$) satisfies the inequality. If it does, leave the side of the line where that point lies unshaded; otherwise, leave the other side unshaded. For the constraint $2x_1 + 3x_2 \leqslant 30$ in the example above, we have $2 \times 0 + 3 \times 0 \leqslant 30$, so the side including the origin should be left unshaded, as in Figure 2.1.

This test fails if the point lies on the constraint bound.

Example 2.1

Draw the feasible region for the table manufacturing problem, where the constraints are modelled as follows.

$$\begin{aligned} 2x_1 + 3x_2 &\leqslant 30 \\ \tfrac{2}{3}x_1 + 2x_2 &\leqslant 16 \\ 5\tfrac{1}{3}x_1 + 4x_2 &\leqslant 64 \\ x_1, x_2 &\geqslant 0 \end{aligned}$$

There is no need to consider the objective of the problem, since the feasible region is determined by the constraints alone.

Solution

We begin by drawing the constraint bounds corresponding to the equalities

$$\begin{aligned} 2x_1 + 3x_2 &= 30 \\ \tfrac{2}{3}x_1 + 2x_2 &= 16 \\ 5\tfrac{1}{3}x_1 + 4x_2 &= 64 \\ x_1 &= 0 \\ x_2 &= 0 \end{aligned}$$

as shown in Figure 2.3(a). We now need to shade the appropriate half-planes. Instead of shading the whole half-plane, it is often easier if we just draw a little shading on the appropriate side of the line, as shown in Figure 2.3(b).

From now on we shall use the method of shading that seems most appropriate for any particular case.

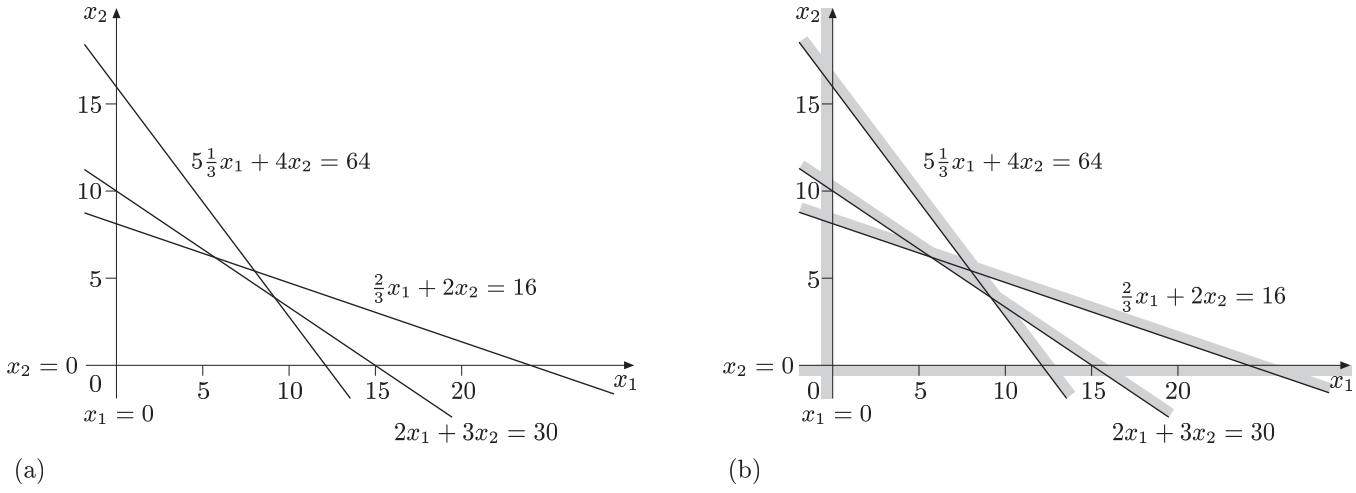


Figure 2.3

The feasible region is the intersection of all the unshaded half-planes, shown unshaded in Figure 2.4.

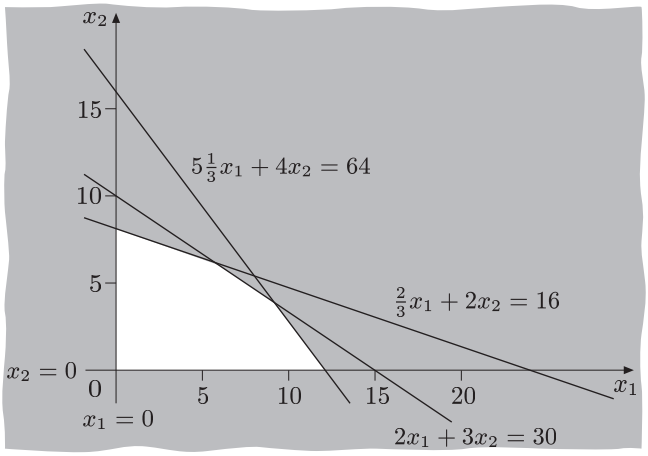


Figure 2.4 ■

Exercise 2.1

Draw the feasible region for the pig feeding problem (Exercise 1.1), whose constraints are modelled as follows.

$$\begin{aligned} 0.6x_1 + \ 0.5x_2 &\geqslant 16 \\ 0.05x_1 + 0.11x_2 &\leqslant \ 3 \\ 0.18x_1 + 0.05x_2 &\geqslant \ 3 \\ x_1, x_2 &\geqslant 0 \end{aligned}$$

The points where two (or more) constraint bounds intersect are known as **vertices**. The line segments joining vertices are known as **edges**. Vertices that lie on the boundary of the feasible region are known as **feasible vertices** or **extreme points**.

Example 2.2

Identify the vertices and feasible vertices of the model in Example 2.1. How many of the edges bound the feasible region?

Solution

There are ten vertices, marked in Figure 2.5. The vertices marked with a white dot are outside the feasible region, while those marked with a black dot lie on its boundary and so are feasible vertices (five in all).

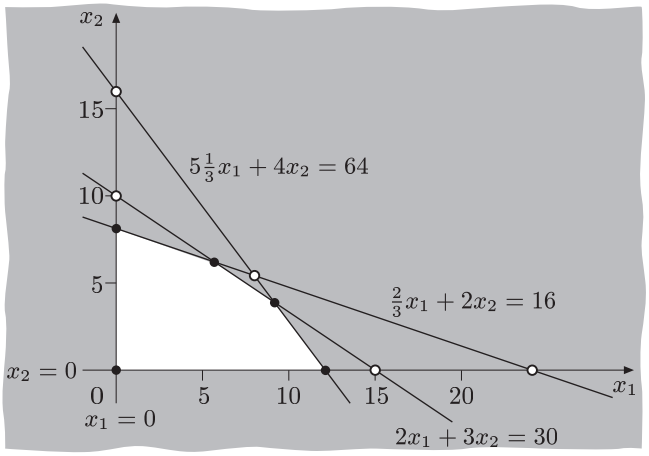


Figure 2.5

Five of the edges bound the feasible region, namely those forming its boundary (joining the feasible vertices). ■

Exercise 2.2

Identify the vertices and feasible vertices of the model in Exercise 2.1.

Types of feasible region

Both the feasible regions we have drawn so far formed finite convex polygons. There are, however, various forms the feasible region can take.

You met convexity in *Unit I.4*.

Bounded feasible region

The feasible region can consist of a bounded (i.e. finite) convex polygon, as we have seen in Figure 2.4, for example. (It is possible, in theory, that the constraints may only be satisfied by a single point, in which case the polygonal feasible region is 'collapsed' to this single point. Similarly, a polygonal feasible region may be 'collapsed' to a line segment.)

You will see examples of such 'collapsed' feasible regions in *Unit II.3*.

Empty feasible region

This will occur when the constraints are mutually contradictory, i.e. there are no values of the variables that can satisfy all the constraints. This may be immediately obvious (if we have, for example, pairs of constraints such as $x_1 - x_2 \leqslant -2$ and $x_1 - x_2 \geqslant 4$), or it may not, as with constraints such as $x_1 - 2x_2 \leqslant 1$, $2x_1 - 5x_2 \geqslant 3$ and $x_1, x_2 \geqslant 0$. Both examples are illustrated in Figure 2.6.

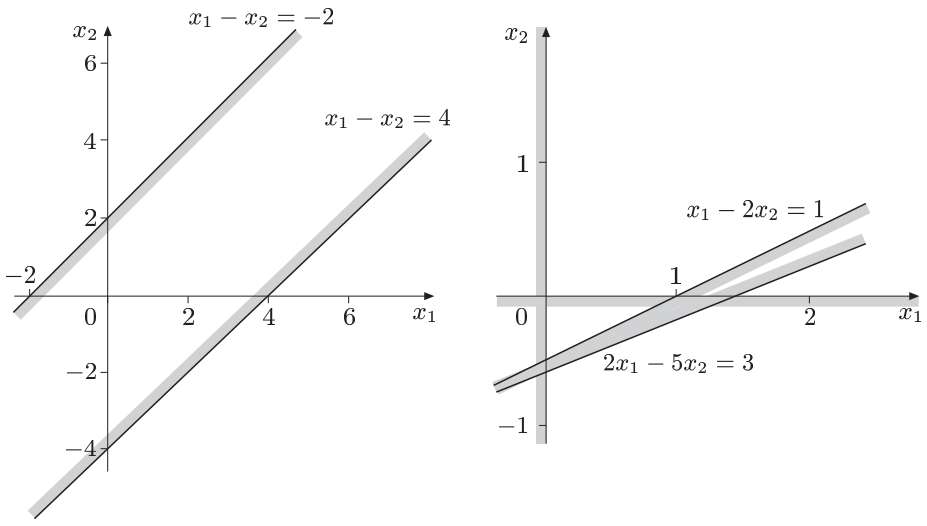


Figure 2.6 Empty feasible regions

Unbounded feasible region

The constraint bounds may not form a bounded convex polygon; rather they may form a feasible region that is infinite. For example, Figure 2.7 illustrates the feasible region for the constraints $x_1 + 2x_2 \geqslant 1, -x_1 + 4x_2 \leqslant 4, x_2 \leqslant 1\frac{1}{2}$ and $x_1, x_2 \geqslant 0$. As you can see, the values x_1 can take are unbounded. In this case we say that the feasible region is *unbounded*.

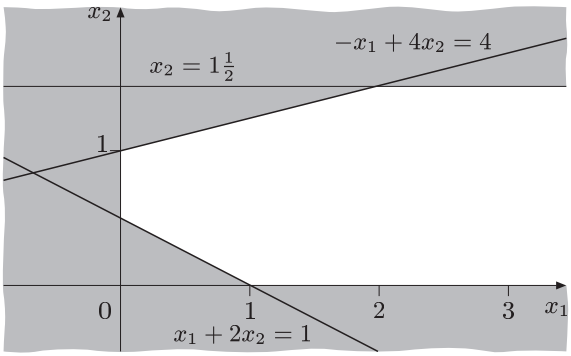


Figure 2.7 An unbounded feasible region

The three cases described above cover all the possible types of feasible region in two dimensions.

Exercise 2.3

Sketch the feasible regions for the following sets of constraints and classify each feasible region as bounded, empty or unbounded.

(a) $x_1 + 2x_2 \leqslant 4$
$3x_1 + 2x_2 \leqslant 5$
$x_1, x_2 \geqslant 0$

(b) $x_1 + x_2 \geqslant 4$
$x_1 \geqslant 2$
$x_1, x_2 \geqslant 0$

(c) $x_1 - 8x_2 \geqslant -10$
$4x_1 - 2x_2 \geqslant -9$
$-x_1 + 4x_2 \leqslant 22$
$x_1, x_2 \geqslant 0$

(d) $x_1 - 8x_2 \geqslant -10$
$4x_1 - 2x_2 \leqslant -9$
$-x_1 + 4x_2 \leqslant 22$
$x_1, x_2 \geqslant 0$

(e) $x_1 + 2x_2 \leqslant 4$
$3x_1 + 2x_2 \leqslant 5$
$7x_1 + 10x_2 \leqslant 21$
$x_1, x_2 \geqslant 0$

(f) $x_1 + 2x_2 \leqslant 4$
$3x_1 + 2x_2 \leqslant 5$
$7x_1 + 10x_2 \geqslant 21$
$x_1, x_2 \geqslant 0$

Degeneracy

In part (e) of Exercise 2.3 the feasible region was the same as in part (a), but there was a third constraint bound passing through one feasible vertex. When more than two constraint bounds pass through a feasible vertex, the vertex is said to be a **degenerate vertex** and the model is said to suffer from **degeneracy**. Although neither the shape of the feasible region nor the solution is affected by degeneracy, it can cause computational difficulties, as we shall see in *Unit II.2*.

Redundancy

Sometimes one or more constraints in a model can be removed without affecting the feasible region. A constraint that can be removed without affecting the feasible region is referred to as a **redundant constraint**. An obvious example occurs in part (b) of Exercise 2.3, where the constraint $x_1 \geqslant 0$ is made redundant by the constraint $x_1 \geqslant 2$. If you look at the feasible region you can see that it is not affected if the constraint $x_1 \geqslant 0$ is removed.

Another example is given by the following set of constraints.

$$\begin{aligned} x_1 + 2x_2 &\leqslant 4 \\ 3x_1 + 2x_2 &\leqslant 5 \\ 3x_1 + 5x_2 &\leqslant 15 \\ x_1, x_2 &\geqslant 0 \end{aligned}$$

The constraint $3x_1 + 5x_2 \leqslant 15$ is redundant, as you can see from Figure 2.8.

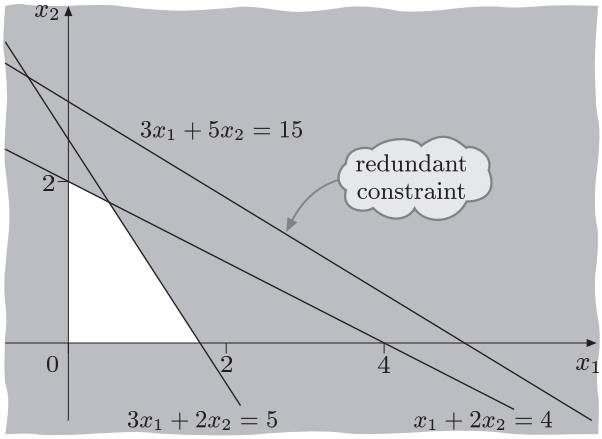


Figure 2.8 A redundant constraint

A model that includes redundant constraints is said to exhibit **redundancy**. A considerable amount of computation can often be saved if redundant non-trivial constraints are spotted and removed before the model is solved.

Exercise 2.4

Look through your solutions to Exercise 2.3 and, for those cases where the feasible region is non-empty, find any redundant constraints.

It is possible to find redundant constraints for an empty feasible region, but there is no point in doing so.

Properties of feasible regions

All feasible regions in linear programming are *convex*. Intuitively, for polygonal regions, this means that there are no 'dents' in them. Moreover, all bounded feasible regions in linear programming are *closed*, i.e. they contain all their boundary points. These properties of feasible regions are fundamental to the theory of the solution of linear programming problems, and we state them as a theorem, whose proof we omit.

Closed convex regions (also referred to as *closed convex sets*) were defined in *Unit I.4*.

Theorem 2.1 Feasible Region Theorem

If the feasible region of a linear programming model is bounded then it is a closed convex set.

This theorem holds even when there is redundancy or degeneracy.

Extension to higher dimensions

Although it is easier to visualize the representations of points in two dimensions, all the ideas we have met in this subsection can readily be extended to higher dimensions. If the standard form of a model has n variables, the vector $[x_1, x_2, \ldots, x_n]^T$ can be represented as a point in n-space, $\mathbb{R}^n$. The constraint bounds become $(n-1)$-dimensional hyperplanes, and each one divides the whole n-space into two halves, with the points satisfying the constraint lying in one of these halves. The feasible region is the intersection of these half-spaces.

A *hyperplane* is the extension to higher dimensions of the concept of a plane in three dimensions (or a line in two dimensions). It is a surface defined by a *linear* equation in n-space.

Definitions

For a linear programming model in standard form with n variables:

- the **feasible region** is the set of points in n-space that satisfy all the constraints, and the points in the feasible region are known as **feasible points**;
- the $(n-1)$-dimensional hyperplanes forming the boundary of the feasible region are **constraint bounds**, and the points on the boundary are **boundary points**;
- the points where n (or more) constraint bounds intersect are **vertices** — those that lie on the boundary of the feasible region are **feasible vertices** or **extreme points**, while those that lie outside the feasible region are **infeasible vertices**;
- the line segments between pairs of vertices, created by the intersection of $n-1$ (or more) constraint bounds, are **edges**;
- if more than n constraint bounds intersect at a feasible vertex, that vertex is said to be a **degenerate vertex** and the model is said to suffer from **degeneracy**;
- if a constraint can be removed from the model without affecting the feasible region, that constraint is said to be a **redundant constraint** and the model is said to suffer from **redundancy**.

These definitions also apply if the model involves minimization and/or if there are $\geqslant$ constraints.

As in the two-dimensional case, the vertices and edges that lie in the feasible region must lie on the boundary of the region. All vertices other than feasible vertices lie outside the feasible region.

The feasible region may be **bounded**, **empty** (if the constraints are mutually contradictory) or **unbounded** (if the values of one or more of the variables are unbounded). Moreover, the Feasible Region Theorem still applies — any bounded feasible region is a closed convex set.

2.2 Graphical solutions

To solve a linear programming model, we need to determine the feasible point or points that optimize the value of the objective function. In the case of two-dimensional problems, we can do this graphically, as the following example illustrates. The solution process corresponds to Stage 3, Do the mathematics, of the mathematical modelling process.

Example 2.3

Solve the table manufacturing problem (Example 1.1) graphically.

Solution

We use the model of Example 1.1. The objective is to

$$\text{maximize } z = 120x_1 + 150x_2$$

and the feasible region is shown in Figure 2.4. To find the point (x_1, x_2) in the feasible region whose values maximize z, we start off by sketching in one or more lines of the form $z = 120x_1 + 150x_2 = \text{constant}$. In order to draw these lines, for any given constant c, we simply need to determine where they cross the x_1- and x_2-axes, by putting $x_2 = 0$ and $x_1 = 0$ in turn in the equation $120x_1 + 150x_2 = c$. We shall choose values of the constant that make the arithmetic simple. We also make use of the fact that, for any given constant c, $120x_1 + 150x_2 = c$ can be rewritten as

$$x_2 = -\tfrac{120}{150}x_1 + \tfrac{c}{150} = -0.8x_1 + \tfrac{c}{150},$$

so all the lines $z =$ constant have the same slope, -0.8, and hence are parallel.

First, we try $z = 120x_1 + 150x_2 = 600$. We can see from Figure 2.9 that this line intersects the feasible region, so there are values (x_1, x_2) that satisfy the constraints and give a value of 600 to the objective function. Let's see if we can improve on this. Let us try $z = 120x_1 + 150x_2 = 1200$. Again the line intersects the feasible region, and we have increased the value of the objective function. As the constant increases, the parallel lines $z =$ constant move away from the origin in the direction shown by the arrow.

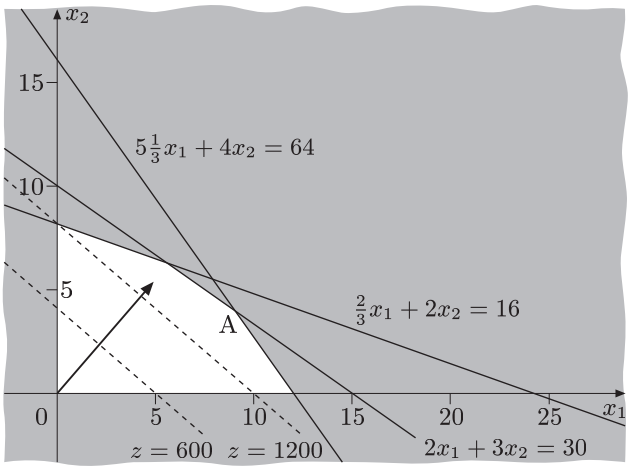


Figure 2.9

The question is, how far can we move the line $z =$ constant in the direction of the arrow and still remain in the feasible region? Now that we have sketched in a couple of lines, and since all the $z =$ constant lines are parallel, we can see that the line $z =$ constant that is furthest from the origin but still has at least one point in common with the feasible region is the line passing through the vertex A in Figure 2.9. The vertex A defines the point (x_1, x_2) for which $z = 120x_1 + 150x_2$ takes its maximum value. It is known as the **optimal vertex** or **optimizer**.

This line can be constructed by drawing one $z =$ constant line and then using a ruler and set square to slide a straight edge parallel to this line until the edge of the feasible region is reached. Very often, though, it is just as easy to judge it by eye having drawn in one or more $z =$ constant lines.

Once the optimal vertex has been found from the graph, the corresponding values of x_1, x_2 and z should be calculated algebraically, *not* estimated from the graph. The optimal vertex A is the point where the line $2x_1 + 3x_2 = 30$ crosses the line $5\frac{1}{3}x_1 + 4x_2 = 64$. So it has coordinates given by the solution of these two simultaneous equations, namely $(9, 4)$. The corresponding value of z is the **optimal value** of z, which we can now calculate as $z = 120x_1 + 150x_2 = 120 \times 9 + 150 \times 4 = 1680$.

Obtaining the optimal vertex and the optimal value of z completes Stage 3, Do the mathematics, of the mathematical modelling process. However, we were asked for a solution to the *problem*, not just the model, so we must now move on to Stage 4 and interpret the results. The interpretation is straightforward: the cooperative should make nine rectangular tables and four circular tables a day for a daily income of £1680. ■

All two-dimensional linear programming models can be solved graphically in this way. The procedure can be summarized as follows.

Procedure 2.1 Graphical solution of two-dimensional linear programming models

(a) Draw the feasible region. If it is empty, stop, as there is no solution.

(b) Draw some $z =$ constant lines and hence determine an optimal vertex, if one exists. If there is no optimal vertex, stop.

(c) Solve the system of linear equations representing the constraint bounds at the optimal vertex, and hence obtain the coordinates of the optimal vertex.

(d) Evaluate z at the optimal vertex, to give the optimal value of z.

You will see shortly that there may be no optimal vertex if the feasible region is unbounded.

In order to obtain accurate values for the coordinates of the optimal vertex, and hence an accurate value for the optimal value of z, it is important to carry out step (c) of Procedure 2.1. You should *not* attempt to estimate the coordinates of the optimal vertex from the graph.

Two other points worth noting arise from Example 2.3.

- The optimal vertex occurred at the intersection of two constraint bounds. This means that the corresponding two constraints, on joinery and final finishing, are at their upper limits, i.e. all 30 person-hours of joinery capacity and all 64 person-hours of final finishing capacity are being used. On the other hand, the vertex A does not lie on the prefinishing constraint bound. So the prefinishing capacity is not fully used, as we can check by putting the values $x_1 = 9$ and $x_2 = 4$ into the left-hand side of the prefinishing constraint. Thus the solution not only tells us the optimal numbers of each type of table and the optimal daily income but also gives us information about the processes involved, enabling us to extend the interpretation of the results. When interpreting the solution to a linear programming model, we shall often want to include such additional information.
- The solution conveniently gave us an exact whole number of each type of table to be made in a day. In this case, a non-integer solution would also have been quite acceptable since we could reasonably assume that tables may be left partly completed one day so that work could continue on them the next. In some problems, however, *only* an integer solution is acceptable (making cakes rather than tables, for example) and for these we shall need other techniques, introduced in *Unit II.3*.

Exercise 2.5

For this exercise, we return to the pig feeding problem of Exercise 1.1.

(a) Use your solution to Exercise 2.1 to find the graphical solution to this model.

(*Hint*: Remember that the pig feeding problem is a minimization, not a maximization, problem.)

(b) Interpret your solution to the model as a solution to the original problem. As part of your interpretation, answer the following two questions.

(i) Will the pigs be fed more than the minimum requirement of protein or amino acids?

(ii) Will they be fed less than the maximum amount of calcium?

2.3 Types of solution

In both the table manufacturing and pig feeding problems the feasible region is a bounded convex polygon and optimizing z defines a unique vertex at the intersection of two constraint bounds. This will not always be the case. The types of solution that may occur are discussed and classified below, in the context of two-dimensional models, before the discussion is extended to higher dimensions at the end of this subsection.

Unique optimizer

Bounded feasible region

We have already seen two examples of a bounded feasible region giving rise to a unique optimal vertex. Figure 2.10(a) shows a similar example, where the objective is to maximize $z = x_1 + x_2$ subject to the constraints $3x_1 + 2x_2 \leqslant 6$, $x_1 + 2x_2 \leqslant 4$ and $x_1, x_2 \geqslant 0$.

In the (trivial) case where the bounded feasible region is a single point, the optimal vertex must lie at this point whatever function is to be optimized.

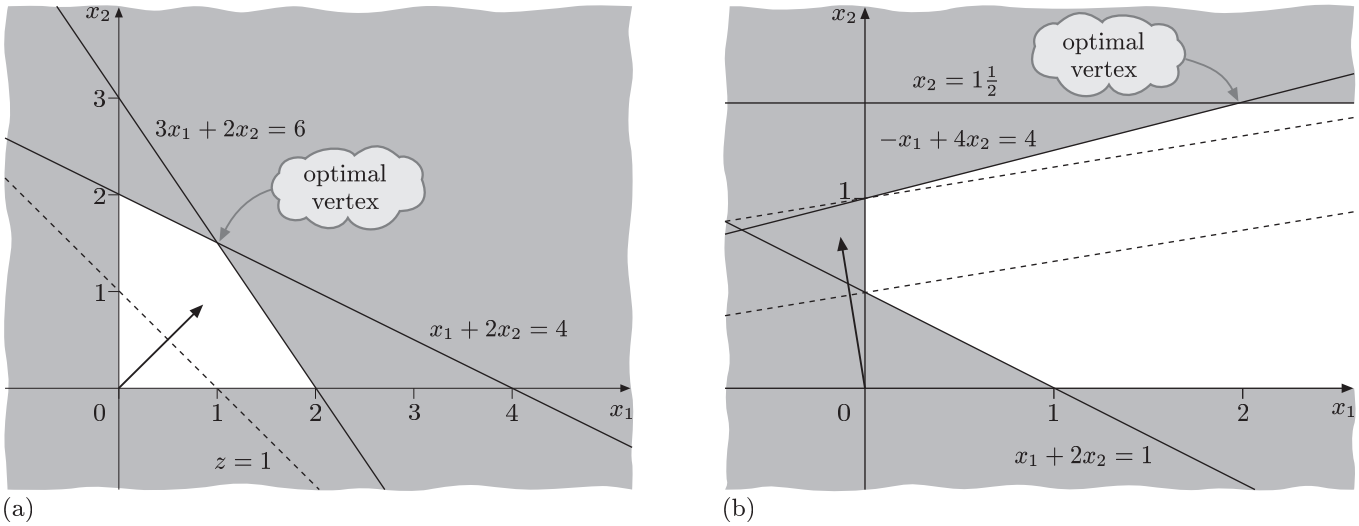


Figure 2.10 A unique optimizer with (a) a bounded feasible region and (b) an unbounded feasible region

Unbounded feasible region

A unique optimal vertex may be obtained from an unbounded feasible region. Figure 2.10(b) shows such an example, where the objective is to maximize $z = -x_1 + 6x_2$ subject to the constraints $x_1 + 2x_2 \geqslant 1$, $-x_1 + 4x_2 \leqslant 4$, $x_2 \leqslant 1\frac{1}{2}$ and $x_1, x_2 \geqslant 0$.

No optimizer

If the feasible region is empty there can be no feasible points, and so obviously no solution to the model. In this case we say that the model is **infeasible**.

In all other cases, where the feasible region is non-empty, the model is said to be **feasible**.

Unbounded solution

It sometimes happens that the solution to a linear programming model with an unbounded feasible region is unbounded, i.e. the objective function can be made as large as we please (if it is to be maximized) or as small as we please (if it is to be minimized) and is not bound by the constraints. For example, Figure 2.11 shows a model where the objective is to maximize $z = x_1 + x_2$ subject to the same constraints as in Figure 2.10(b). Here, x_2 cannot exceed $1\frac{1}{2}$ but x_1 can be made as large as we please; so there is no upper limit for $z = x_1 + x_2$. When the solution is unbounded we also refer to the model as **unbounded**.

Of course, if the feasible region is bounded then the solution must be too.

If the solution to a feasible linear programming model is bounded, then we refer to the model as **bounded**.

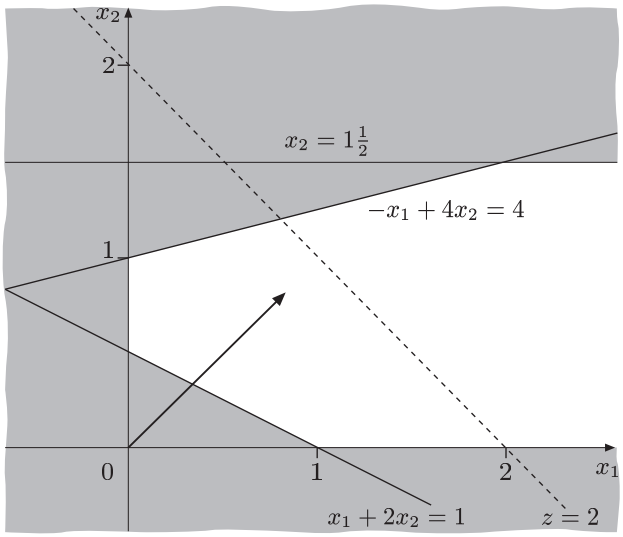


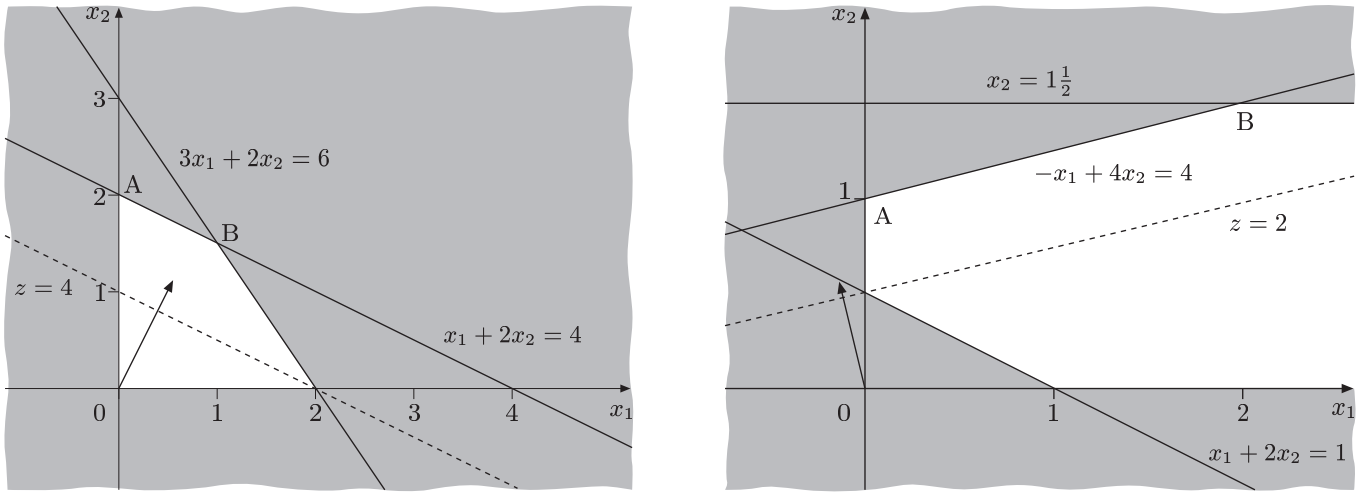
Figure 2.11 An unbounded solution

Multiple optimizers

It can happen that, if the lines $z =$ constant lie parallel to one of the constraint bounds, the objective function may achieve its optimal value along a whole edge of the feasible region rather than at a single vertex.

Bounded feasible region

Figure 2.12(a) shows the case where the objective is to maximize $z = 2x_1 + 4x_2$ subject to the same constraints as in Figure 2.10(a). The feasible region is thus the same as in Figure 2.10(a) but the lines $z =$ constant are now parallel to the constraint bound $x_1 + 2x_2 = 4$ and so z will take its optimal value (of 8) at all points along the edge AB.



(a) (b)

Figure 2.12 Multiple optimizers with (a) a bounded feasible region and (b) an unbounded feasible region

Unbounded feasible region

Figure 2.12(b) shows the case where the objective is to maximize $z = -x_1 + 4x_2$ subject to the same constraints as in Figures 2.10(b) and 2.11. The lines $z =$ constant are parallel to the constraint bound $-x_1 + 4x_2 = 4$ and again we have optimizers at all points along AB.

Exercise 2.6

Solve each of the following linear programming models graphically and classify the type of solution in each case.

You have already sketched the feasible regions for these models in Exercise 2.3.

(a) maximize $z = x_1 + x_2$
subject to
$x_1 + 2x_2 \leqslant 4$
$3x_1 + 2x_2 \leqslant 5$
$x_1, x_2 \geqslant 0$

(b) (i) minimize $z = x_1 + x_2$
(ii) maximize $z = x_1 + x_2$
subject to
$x_1 + x_2 \geqslant 4$
$x_1 \geqslant 2$
$x_1, x_2 \geqslant 0$

(c) maximize $z = -2x_1 + x_2$
subject to
$x_1 - 8x_2 \geqslant -10$
$4x_1 - 2x_2 \geqslant -9$
$-x_1 + 4x_2 \leqslant 22$
$x_1, x_2 \leqslant 0$

(d) maximize $z = x_1 + x_2$
subject to
$x_1 - 8x_2 \geqslant -10$
$4x_1 - 2x_2 \leqslant -9$
$-x_1 + 4x_2 \leqslant 22$
$x_1, x_2 \geqslant 0$

(e) (i) maximize $z = 3\frac{1}{2}x_1 + 5x_2$
(ii) maximize $z = 3x_1 + 2x_2$
subject to
$x_1 + 2x_2 \leqslant 4$
$3x_1 + 2x_2 \leqslant 5$
$7x_1 + 10x_2 \leqslant 21$
$x_1, x_2 \geqslant 0$

(f) maximize $z = 3x_1 + 2x_2$
subject to
$x_1 + 2x_2 \leqslant 4$
$3x_1 + 2x_2 \leqslant 5$
$7x_1 + 10x_2 \geqslant 21$
$x_1, x_2 \geqslant 0$

Extension to higher dimensions

We have seen in our two-dimensional examples that, where a finite solution exists, it always occurs at a *vertex* of the feasible region, i.e. at a feasible vertex. This result is true in general and extends to more than two dimensions. We therefore state this important result as a second theorem, whose proof we again omit.

This includes the case of multiple optimizers, where the optimal value of the objective function occurs at two vertices (and at the points on the edge joining them).

> **Theorem 2.2 Optimal Vertex Theorem**
>
> If a linear programming model has a finite solution, then that solution is attained at a feasible vertex.

In light of this theorem, we can make the following general definitions, which apply to models of any dimension.

> **Definitions**
>
> The value of the objective function at the solution to a linear programming model is referred to as the **optimal value of the objective function**.
>
> A feasible vertex at which the objective function of a linear programming model achieves its optimal value is referred to as an **optimal vertex** or **optimizer**.

A model of any dimension may have *no optimizer*, a *unique optimizer*, an *unbounded solution* or *multiple optimizers*. It is worth noting that, for dimensions greater than two, multiple optimizers can occur when the hyperplane $z =$ constant coincides with *more than one* edge of the feasible region (rather than with just one, as in the two-dimensional case). Such situations can be hard to recognize.

The properties of *feasibility*, *infeasibility*, *boundedness* and *unboundedness*, when applied to a linear programming model, also extend naturally to higher dimensions.

2.4 Graphical representation of canonical form

We have seen how to solve a two-dimensional linear programming model graphically. We shall now see how the canonical form of a linear programming model and its solution can be interpreted graphically.

Example 2.4

The canonical form of the model for the table manufacturing problem (Example 1.1) is:

$$\text{maximize } z = 120x_1 + 150x_2$$

subject to

$$\begin{aligned}
2x_1 + 3x_2 + x_3 &= 30 && \text{(joinery)}\\
\tfrac{2}{3}x_1 + 2x_2 + x_4 &= 16 && \text{(prefinishing)}\\
5\tfrac{1}{3}x_1 + 4x_2 + x_5 &= 64 && \text{(final finishing)}
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5 \geqslant 0$$

Interpret this form of the model, and its solution, graphically.

Solution

In Example 2.1, for the joinery constraint, we drew the constraint bound $2x_1 + 3x_2 = 30$ and shaded one side of the line; the unshaded half-plane was where $2x_1 + 3x_2 \leqslant 30$. This line now corresponds to $x_3 = 0$ and the unshaded half-plane to $x_3 \geqslant 0$. Similarly, the constraint bounds for the other constraints correspond to each of the slack variables taking the value zero, and the unshaded half-planes to the non-negativity constraints on the slack variables. The two constraint bounds given by the x_1-axis and x_2-axis correspond to $x_2 = 0$ and $x_1 = 0$ respectively. Thus the graphical representation of the canonical form of the model gives the same picture as before but with a different interpretation of the lines and vertices, as in Figure 2.13. Here each line corresponds to one of the variables being zero and each vertex corresponds to two of the variables being zero.

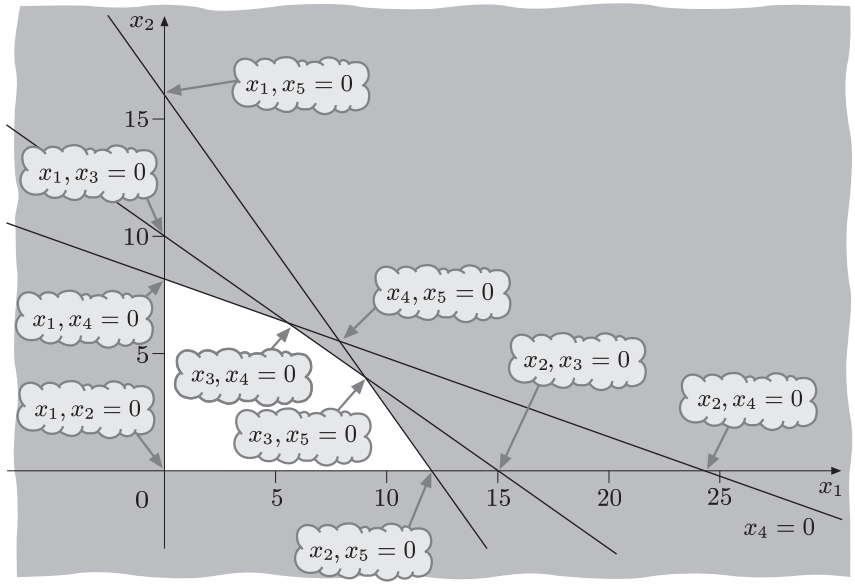


Figure 2.13

The coordinates of any vertex can be calculated by setting the appropriate two variables to zero in the constraint equations. For example, to find the coordinates of the (optimal) vertex where $x_3 = 0$ and $x_5 = 0$, we solve the system of linear equations

$$\begin{aligned} 2x_1 + 3x_2 \qquad &= 30 \\ \tfrac{2}{3}x_1 + 2x_2 + x_4 &= 16 \\ 5\tfrac{1}{3}x_1 + 4x_2 \qquad &= 64 \end{aligned}$$

to obtain $x_1 = 9$, $x_2 = 4$, $x_4 = 2$. We have thus obtained the same solution as before, with the additional information that $x_4 = 2$. This can be interpreted as telling us that, at the solution, there are two spare person-hours of prefinishing capacity, and so provides useful additional information. ■

We saw earlier (Subsection 2.2) that, for an n-dimensional linear programming model in standard form, the vertices are where n (or more) constraint bounds intersect. Generalizing part of the discussion in Example 2.4, we can also conclude that, for a model in canonical form with m constraints and $n + m$ variables, the vertices are where n (or more) of the variables are zero. Also, each constraint bound is an $(n - 1)$-dimensional hyperplane on which one of the variables is zero. To find a given vertex, we can set n of the variables to zero in the constraint equations and then solve the resulting system of m equations in m unknowns.

Exercise 2.7

The canonical form of the model for the pig feeding problem (Exercise 1.1) is as follows.

$$\begin{aligned}
&\text{maximize } z = -15x_1 - 12x_2 \\
&\text{subject to} \\
&0.6x_1 + 0.5x_2 - x_3 = 16 \quad \text{(protein)} \\
&0.05x_1 + 0.11x_2 + x_4 = 3 \quad \text{(calcium)} \\
&0.18x_1 + 0.05x_2 - x_5 = 3 \quad \text{(amino acids)} \\
&x_1, x_2, x_3, x_4, x_5 \geqslant 0
\end{aligned}$$

Interpret this form of the model and its solution graphically.

You solved this model graphically in Exercise 2.5.

End-of-section Exercises

Exercise 2.8

In Exercise 1.16 you modelled the shampoo manufacturing problem as

$$\begin{aligned}
&\text{minimize } z = 15x_1 + 22.5x_2 \\
&\text{subject to} \\
&0.2x_1 + 0.05x_2 \geqslant 1 \\
&0.2x_1 + 0.5x_2 \geqslant 3 \\
&0.2x_1 + 0.3x_2 \leqslant 2.5 \\
&x_1 + x_2 \leqslant 10 \\
&x_1, x_2 \geqslant 0
\end{aligned}$$

(a) Solve this model graphically.

(b) Deduce the values of the variables x_3, x_4, x_5 and x_6 introduced in the solution to Exercise 1.16 to obtain the canonical form of the model.

(c) Interpret your solution as a solution to the original problem.

3 The simplex method

Since the graphical method of solving linear programming models cannot be used when there are more than two variables, we need a new method if we are to tackle more complex problems. This section introduces a method for solving linear programming models, expressed in canonical form, that can be used however many variables there are: the *simplex method*. The method is discussed algebraically in this section; a matrix formulation is given in Section 4. Throughout this section, and most of the next, we assume that the models do not suffer from degeneracy; we shall consider degenerate models in Subsection 4.4. We shall also assume that the general form of the model contains no equality constraints; models with equality constraints will be considered in *Unit II.2*.

The method is extended in *Unit II.2*.

3.1 Introduction to the simplex method

In Section 2 we stated, though we did not prove, that the solution to a bounded linear programming model is always achieved at a feasible vertex. Using this result, a sledgehammer method for solving any linear programming model might be to determine all the vertices, find the ones that are feasible, evaluate the objective function at each of them, and select the one which gives the optimal value for the objective function. However, this approach has serious computational problems. To find each vertex, we can do as we did in Subsection 2.4: for a model in canonical form with m constraints equations and $n + m$ variables, we can set n of the variables to zero and solve a system of m equations in m unknowns. However, we would need to do this for every possible combination of n variables to determine all the vertices, and there are

$$\binom{n+m}{n} = \frac{(n+m)!}{m!\,n!}$$

ways of choosing n of the $n + m$ variables. Thus, for example, if the standard form of the model has 16 variables and 20 non-trivial constraints, the canonical form will have 36 variables and 20 equations. Just determining the vertices for such a small problem using this approach will require the solution of $36!/(20!16!) = 7\,307\,872\,110$ systems of 20 equations in 20 unknowns. For this reason the sledgehammer approach is never used. Instead, the standard method for solving linear programming problems is the **simplex method**.

See Theorem 2.2.

The simplex method was first developed by George B. Dantzig in 1947 while he was working as a mathematical adviser at the Pentagon in the United States.

The simplex method is one of the class of computer techniques that are grouped together under the name of **hill-climbing methods**. The outline of all hill-climbing methods is the same.

(a) Find a feasible point, i.e. a point that satisfies the constraints of the problem.
(b) Can you find a neighbouring feasible point that is higher than the current point? If so, move to the higher point and repeat this step.
(c) If all neighbouring feasible points are lower than the current feasible point, stop.

In the simplex method, the measure of height used is the value of the objective function and the points around which we move are the feasible vertices. We always move to an *adjacent* feasible vertex, where two vertices are defined to be **adjacent** if they are connected by an edge.

General hill-climbing methods suffer from a major difficulty. In some non-linear problems, it is possible to have a **local optimizer** (i.e. a point which is higher than all its neighbours) that is not the overall optimizer, or **global optimizer**. This is illustrated in Figure 3.1.

You will have met the ideas of local optimizers (local maximizers and minimizers) and global optimizers (global maximizers and minimizers) before, in the context of functions of one variable, in MST209 or M208 for example, and in *Unit I.5* of this course. In Block III, you will meet non-linear optimization problems that have local optimizers which are not global optimizers.

Figure 3.1 A local maximizer and the global maximizer for a non-linear problem

However, for this to happen there must be a concavity between the optimizers, as in Figure 3.1. In the case of linear programming problems, we have seen that the feasible region is always convex, and this, together with the linearity of the objective function, precludes the possibility of having a local optimizer that is not also a global optimizer. We thus have the following result, which is stated without proof.

Theorem 3.1

All local optimizers of a linear programming model are also global optimizers.

The simplex method is based on the canonical form of a linear programming model and, as mentioned above, involves moving repeatedly from one feasible vertex to an adjacent one. To have a clear idea of how the simplex method works, it is going to be necessary to have a clear idea of how vertices in general, and feasible vertices in particular, are represented in canonical form. You were introduced to this topic in Subsection 2.4. Let us recapitulate, in the context of the two-dimensional example that we looked at there: the table manufacturing problem (Example 1.1). The canonical form of this model is as follows.

maximize $z = 120x_1 + 150x_2$

subject to

$$\begin{aligned} 2x_1 + 3x_2 + x_3 &= 30 && \text{(joinery)} \\ \tfrac{2}{3}x_1 + 2x_2 + x_4 &= 16 && \text{(prefinishing)} \\ 5\tfrac{1}{3}x_1 + 4x_2 + x_5 &= 64 && \text{(final finishing)} \end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5 \geqslant 0$$

The feasible region and constraint bounds are shown in Figure 3.2.

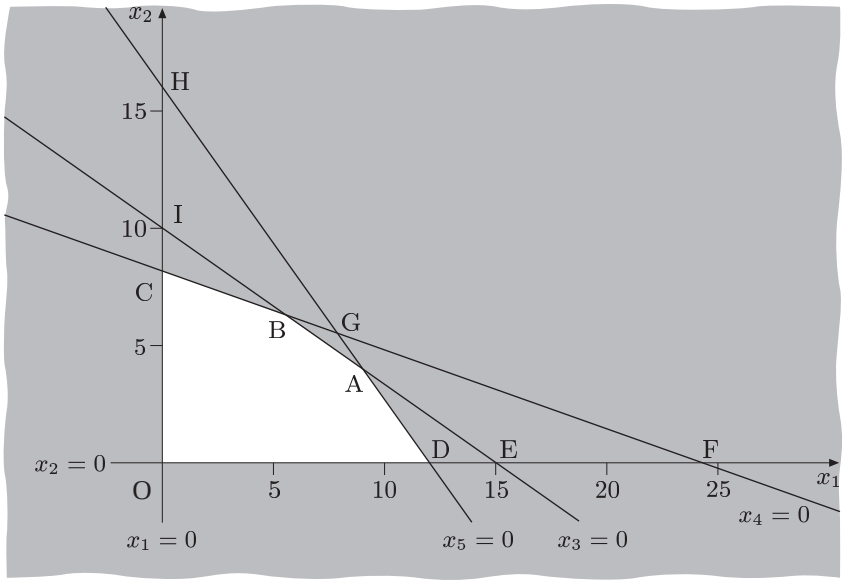


Figure 3.2

You may recall, from Subsection 2.4, that each constraint bound corresponds to setting one of the variables to zero, as indicated in the figure. Also, in this two-dimensional example, each vertex occurs where two constraint bounds meet, i.e. where two of the variables are zero. We shall now introduce some new terminology, which will prove helpful when using the simplex method.

Definition

Given a linear programming model in canonical form:

- a feasible vertex is referred to as a **basic feasible point** and an infeasible vertex as a **basic infeasible point**;
- if the model is non-degenerate, the variables that are non-zero at a vertex are referred to as **basic variables**, while those that are zero are **non-basic variables**;
- a constraint whose constraint bound passes through a vertex is said to be **active** at that vertex, otherwise it is **inactive** at that vertex.

We shall refine our definition of basic and non-basic variables later to take account of degeneracy.

Thus, in Figure 3.2, the vertices are the points labelled O, A, B, C, D, E, F, G, H and I. The feasible points are all the points in the feasible region (including those on its boundary). The basic feasible points are those labelled O, A, B, C and D; the basic infeasible points are those labelled E, F, G, H and I. At the point O, the basic variables are x_3, x_4 and x_5, while the non-basic variables are x_1 and x_2. At A, the basic variables are x_1, x_2 and x_4, while the non-basic ones are x_3 and x_5. At O, the active constraints are those corresponding to $x_1 = 0$ and $x_2 = 0$, namely $x_1 \geqslant 0$ and $x_2 \geqslant 0$, and the inactive constraints are the remaining ones: $x_3 \geqslant 0$, $x_4 \geqslant 0$ and $x_5 \geqslant 0$, or equivalently $2x_1 + 3x_2 \leqslant 30$, $\frac{2}{3}x_1 + 2x_2 \leqslant 16$ and $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$ respectively. At A, the active constraints are those corresponding to $x_3 = 0$ and $x_5 = 0$, namely $2x_1 + 3x_2 \leqslant 30$ and $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$, and the inactive constraints are the remaining ones: $x_1 \geqslant 0$, $x_2 \geqslant 0$ and $x_4 \geqslant 0$ (or equivalently $\frac{2}{3}x_1 + 2x_2 \leqslant 16$). Notice that, at a vertex, the non-basic variables correspond to active constraints.

Exercise 3.1

Determine the basic and non-basic variables and the active and inactive constraints at the points C and G in Figure 3.2.

The simplex method starts from a basic feasible point, i.e. a feasible vertex. If the origin (the point O in Figure 3.2) is a basic feasible point, it starts there. In our example, O has two adjacent feasible vertices, D and C, as there are two active constraints, $x_1 \geqslant 0$ and $x_2 \geqslant 0$. To move to an adjacent feasible vertex, D say, we move along the edge OD. Once we leave O, $x_1 \geqslant 0$ is no longer active, and the only active constraint is $x_2 \geqslant 0$. Once we reach D, a new constraint, $x_5 \geqslant 0$ (i.e. $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$), becomes active. Thus as we leave one vertex, one variable becomes non-zero and one active constraint becomes inactive. As we reach an adjacent feasible vertex, a different variable becomes zero, and a new constraint becomes active.

In this unit we shall only apply the simplex method to problems where the origin is a feasible vertex. We shall tackle problems where it is not in *Unit II.2*.

Exercise 3.2

Describe what happens, in terms of basic and non-basic variables and active and inactive constraints, if we move from D to the adjacent feasible vertex A in Figure 3.2.

The simplex method works by keeping track of the basic and non-basic variables and active and inactive constraints as it moves from basic feasible point to basic feasible point, as we shall see in Subsection 3.2.

For a general non-degenerate n-dimensional model, whose canonical form has m non-trivial constraints and $n + m$ variables (the last m of which are slack variables), each vertex occurs at the intersection of n $(n-1)$-dimensional hyperplanes. On each of these, $x_j = 0$ for some variable x_j ($j \in \{1, 2, \ldots, n+m\}$). Thus, at each vertex, n variables are non-basic (zero) and the remainder are basic (non-zero). The simplex method starts from a basic feasible point (feasible vertex), the origin if this is feasible. The origin is the point at which $x_j = 0$ for $j = 1, 2, \ldots, n$ and so at which $x_j > 0$ for $j = n+1, \ldots, n+m$, i.e. at which all the slack variables are non-zero; for this reason the origin is often referred to as the *all-slack point*. (Formally, the **all-slack point** is the vertex at which all the main variables are zero.) The simplex method moves from one basic feasible point to an adjacent one along an edge, which is the intersection of $n-1$ hyperplanes, and thus along which $n-1$ of the variables are non-basic (zero), so that $n-1$ of the constraints are active. As we leave one basic feasible point to move along an edge, we leave one of the hyperplanes (representing a constraint bound), and so one of the variables becomes basic (non-zero), and that constraint becomes inactive. Then, when we reach the next basic feasible point, we meet a new hyperplane (representing a new constraint bound), and so one other variable becomes non-basic (zero), and the new constraint becomes active. The method continues moving from basic feasible point to adjacent basic feasible point until it reaches an optimizer. We shall see how the method determines which adjacent basic feasible point to move to and when it has reached an optimizer in the next subsection.

Exercise 3.3

The canonical form of the model for the gardening problem (Exercise 1.15) is:

maximize $z = 3x_1 + 5x_2 + 4x_3$

subject to

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 &= 600 && \text{(cutting)}\\
x_2 + x_3 + x_5 &= 400 && \text{(potting off)}\\
x_1 + x_2 + x_6 &= 300 && \text{(potting on)}\\
x_1 + x_7 &= 80 && \text{(marketing)}
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geqslant 0$$

(a) Describe briefly, along the lines of the paragraph preceding this exercise, how the simplex method goes about solving this three-dimensional gardening problem.

(b) Find the values of the four basic variables and the value of z at each of the following vertices:

(i) the origin (where $x_1 = x_2 = x_3 = 0$);

(ii) the intersection of $x_1 = 0$, $x_3 = 0$ and $x_6 = 0$;

(iii) the intersection of $x_1 = 0$, $x_5 = 0$ and $x_6 = 0$;

(iv) the intersection of $x_5 = 0$, $x_6 = 0$ and $x_7 = 0$.

You can think of the simplex method as an intelligent beetle that can move only along the edges of the feasible region of a linear programming model. The beetle repeats the following three steps until it reaches an optimal vertex:

(a) it stands at a feasible vertex (initially at the origin if this is feasible);

(b) it decides which edge to move along (if no edge improves its position, it is at an optimal vertex);

(c) it crawls along the chosen edge until it arrives at another feasible vertex.

In the context of the canonical form of a linear programming model, 'improves' here means 'increases the value of the objective function'.

3.2 An algebraic description

We shall give a formal matrix formulation of the simplex method in the next section. Here we provide an algebraic description in the context of the gardening problem.

Example 3.1

Solve the gardening problem (Exercise 1.15) by the simplex method, and interpret the solution.

Solution

The canonical form of the model for the problem is:

$$\text{maximize } z = 3x_1 + 5x_2 + 4x_3$$

subject to

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 &= 600 && \text{(cutting)}\\
x_2 + x_3 + x_5 &= 400 && \text{(potting off)}\\
x_1 + x_2 + x_6 &= 300 && \text{(potting on)}\\
x_1 + x_7 &= 80 && \text{(marketing)}
\end{aligned}$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geqslant 0$$

Iteration 1, Step (a): determine a basic feasible point

By putting $x_1 = x_2 = x_3 = 0$ in the four constraint equations we can readily check that the all-slack point (the origin) is feasible. The values of the basic variables and the objective function at this point are $x_4 = 600$, $x_5 = 400$, $x_6 = 300$, $x_7 = 80$ and $z = 0$, as we found in Exercise 3.3. Thus a basic feasible point for starting the simplex method is $\mathbf{x} = [0, 0, 0, 600, 400, 300, 80]^T$, at which $z = 0$. At this point, the active constraints correspond to the non-basic variables x_1, x_2 and x_3. The subscripts of the basic variables form the **basis list**, which we shall write as $\boxed{4 \mid 5 \mid 6 \mid 7}$.

Iteration 1, Step (b): choose the variable to become basic

We now want to move along an edge of the feasible region to an adjacent basic feasible point. Since the origin is where the three planes defined by $x_1 = 0$, $x_2 = 0$ and $x_3 = 0$ meet, there are three edges we can move along: those defined by $x_1 = 0$ and $x_2 = 0$, $x_1 = 0$ and $x_3 = 0$, and $x_2 = 0$ and $x_3 = 0$. If we move along the edge defined by $x_2 = 0$ and $x_3 = 0$, say, x_1 becomes non-zero, i.e. x_1 becomes basic. Similarly, if we choose to move along the edges defined by $x_1 = 0$ and $x_3 = 0$ or by $x_1 = 0$ and $x_2 = 0$, x_2 or x_3 will become basic. We choose the direction which causes the most rapid increase in the value of the objective function, where 'most rapid' means 'greatest increase in z per unit length along the edge'. To find this direction we consider the effect on the objective function of increasing the value of each of the non-basic variables by one — this will give the increase in z per unit length in each direction. Now

These edges correspond to the x_3-axis, x_2-axis and x_1-axis respectively.

This is not the only way of choosing the variable to become basic. We could choose any direction that increases the value of z. The method we describe is one that generally works well in practice.

$$z = 3x_1 + 5x_2 + 4x_3,$$

and $z = 0$ at the origin. If we increase x_1 by one, z will increase by 3; if we increase x_2 by one, z will increase by 5; and if we increase x_3 by one, z will increase by 4. Clearly the greatest rate of improvement is made by increasing x_2, i.e. by moving along the edge where $x_1 = 0$ and $x_3 = 0$. So we choose x_2 as the variable that is to become basic. All we had to do was to choose the non-basic variable in z with the greatest coefficient.

Recall that the coefficients of the objective function in a maximization problem can be regarded as *negative costs*. In this case we have

Costs are deducted from profit, whereas negative costs increase profit.

$$z = -d_1x_1 - d_2x_2 - d_3x_3,$$

where $d_1 = -3$, $d_2 = -5$ and $d_3 = -4$. The coefficients d_j, when z is expressed as a negative linear combination of the non-basic variables, are known as **reduced costs**, and the greatest increase in z is made by choosing the direction corresponding to the most negative reduced cost, d_2.

The reasons for writing z in this way when using the simplex method are historical.

Iteration 1, Step (c): determine the constraint to become active

As we begin to move along the edge where x_1 and x_3 are zero (and the constraints corresponding to $x_1 = 0$ and $x_3 = 0$ are active), all the other variables are non-zero and changing in value. We now have to determine how far we can go before we reach a basic feasible point. This means finding which constraint bound we meet first when moving along the edge where $x_1 = x_3 = 0$ and x_2 increases from zero. In other words, we need to determine which constraint becomes active first, i.e. which of the variables x_4, x_5, x_6 or x_7 becomes zero first. To do this, we shall consider the effect on each of the variables x_4, x_5, x_6 and x_7 of increasing x_2 from zero by an amount θ while keeping $x_1 = x_3 = 0$. The constraint equations with $x_2 = \theta$ and $x_1 = x_3 = 0$ become

We cannot allow any of the variables to become negative as this would imply a basic infeasible point.

$$\begin{aligned} \theta + x_4 &= 600 \\ \theta + x_5 &= 400 \\ \theta + x_6 &= 300 \\ x_7 &= \ 80 \end{aligned}$$

Note that $\theta = 0$ gives the solution at the current basic feasible point.

giving $x_4 = 600 - \theta$, $x_5 = 400 - \theta$, $x_6 = 300 - \theta$ and $x_7 = 80$. As θ increases from zero, x_6 becomes zero first, when $\theta = 300$. Thus the constraint corresponding to $x_6 = 0$ becomes active first, and x_6 becomes non-basic. Replacing 6, the subscript of x_6, in the basis list by 2, the subscript of x_2 (which is now basic), the new basis list is $\boxed{4 \mid 5 \mid 2 \mid 7}$.

This is the potting-on constraint, $x_1 + x_2 \leqslant 300$.

While the ordering of the basis list is not important from a conceptual point of view, the order given corresponds to the order in which data is stored in the matrix formulation.

Iteration 2, Step (a): determine the new basic feasible point

The new basic feasible point is where $\theta = 300$. With $\theta = 300$, from step (c) of Iteration 1 we have $x_2 = 300$, $x_4 = 300$, $x_5 = 100$ and $x_7 = 80$. Thus the new basic feasible point is $\mathbf{x} = [0, 300, 0, 300, 100, 0, 80]^T$, at which $z = 1500$.

Notice that we only need to calculate the values of the basic variables in step (a) since, by definition, the non-basic variables are all zero-valued.

Iteration 2, Step (b): choose the variable to become basic

We now need to consider the effect on the objective function of increasing each of the non-basic variables, x_1, x_3 and x_6, in turn. This is not quite as simple as it was the first time round because, in order to do so, we need to express z solely in terms of the non-basic variables. We can do this by expressing x_2 in terms of x_1, x_3 and x_6. From the potting-on constraint equation we have

$$x_2 = 300 - x_1 - x_6.$$

Thus we have

$$\begin{aligned} z &= 3x_1 + 5(300 - x_1 - x_6) + 4x_3 \\ &= 1500 - 2x_1 + 4x_3 - 5x_6. \end{aligned}$$

So we can see that increasing x_1 or x_6 will reduce the value of z, but increasing x_3 will increase the value of z. So there is only one choice: x_3 should become basic.

Using the reduced cost terminology introduced above, we have $d_1 = 2$, $d_3 = -4$ and $d_6 = 5$, so we choose the variable, x_3, corresponding to the most negative (in this case the only negative) reduced cost, d_3, to become basic.

Iteration 2, Step (c): determine the constraint to become active

We now want to find, with $x_1 = x_6 = 0$ and as x_3 increases from zero, which of the variables x_2, x_4, x_5 or x_7 becomes zero first, i.e. which of the corresponding constraints becomes active first. We put $x_1 = x_6 = 0$ and $x_3 = \theta$, then let θ increase from zero to see the effect on the other variables. We have, from the constraint equations,

$$\begin{aligned} x_2 + \theta + x_4 &= 600 \\ x_2 + \theta + x_5 &= 400 \\ x_2 &= 300 \\ x_7 &= 80 \end{aligned}$$

giving $x_2 = 300$, $x_4 = 300 - \theta$, $x_5 = 100 - \theta$, $x_7 = 80$. Thus, x_5 will become zero first, when $\theta = 100$, so the constraint corresponding to $x_5 = 0$ becomes active and x_5 becomes non-basic. Replacing 5, the subscript of x_5, by 3, the subscript of x_3, the new basis list is $\boxed{4 \mid 3 \mid 2 \mid 7}$.

This is the potting-off constraint, $x_2 + x_3 \leqslant 400$.

Exercise 3.4

Perform the third iteration of the simplex method for the gardening problem.

Iteration 4, Step (a): determine the new basic feasible point

With $\theta = 80$, from step (c) of Iteration 3 we have $x_1 = 80$, $x_2 = 220$, $x_3 = 180$ and $x_4 = 120$. Thus $\mathbf{x} = [80, 220, 180, 120, 0, 0, 0]^T$ is the new basic feasible point, at which $z = 2060$.

Iteration 4, Step (b): choose the variable to become basic

We need to express z in terms of the non-basic variables x_5, x_6 and x_7. We have, from Iteration 3,

$$z = 1900 + 2x_1 - 4x_5 - x_6.$$

From the marketing constraint we have

$$x_1 = 80 - x_7,$$

and using this to eliminate x_1 from z gives

$$z = 2060 - 4x_5 - x_6 - 2x_7.$$

Now all the reduced costs are positive. This means that increasing x_5, x_6 or x_7 will *reduce* the value of objective function. Thus the objective function cannot be increased in any direction, and so we must have reached an optimal vertex. The solution is therefore $x_1 = 80$, $x_2 = 220$, $x_3 = 180$, $x_4 = 120$, $x_5 = x_6 = x_7 = 0$ and $z = 2060$.

Interpreting these results in terms of the original problem, the head gardener should treat enough plants to enable 80 azaleas, 220 rhododendrons and 180 camellias to be sold per day, for a daily income of £2060. This will involve cutting 480 plants per day, potting off 400 plants per day and potting on 300 plants per day. There will be spare capacity for cutting 120 plants a day. ■

Procedure 3.1 Simplex method for solving linear programming models (algebraic version)

Given a linear programming model in canonical form with $n + m$ variables and m constraint equations, start from a basic feasible point (the all-slack point if possible), where the non-basic variables are zero and the basic variables are defined by a current basis list B, and repeat the following steps.

The initial basis list, corresponding to the origin, is usually

$n+1$	$n+2$	...	$n+m$

(a) *Determine the current basic feasible point*

Set the non-basic variables to zero, determine the values of the basic variables, and hence determine the current value of the objective function.

(b) *Choose the variable to become basic*

Find the variable that will cause the greatest rate of increase in z by finding the effects of increasing the value of each non-basic variable. This entails expressing z in terms of the non-basic variables, giving

$$z = k - \sum_{j \notin B} d_j x_j,$$

You will see in the next section that the constant k is the current value of the objective function.

where the coefficients d_j are the *reduced costs*. If all the reduced costs are non-negative, stop, as the current basic feasible point is a solution; otherwise, choose the variable x_q with the most negative reduced cost to become basic.

(c) *Determine the constraint to become active*

Find which variable becomes zero first, as x_q increases from zero, by using the constraint equations and the fact that the non-basic variables are zero to express each basic variable in terms of $x_q = \theta$. If θ can be increased indefinitely without any basic variable becoming zero, stop. Otherwise, find the smallest non-negative value of θ at which one of the basic variables becomes zero. That basic variable, x_p say, is the variable that becomes non-basic, and whose corresponding constraint becomes active. Update the basis list, replacing p by q.

If θ can be increased indefinitely without any basic variable becoming zero, we can move in the direction of increasing x_q without encountering another constraint bound. In other words, the model must be *unbounded*.

Exercise 3.5

Write the following linear programming model in canonical form:

$$\text{maximize } z = 2x_1 - 5x_2 + 3x_3$$

subject to

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &\leqslant 12 \\ 3x_1 - 2x_2 + x_3 &\leqslant 8 \\ 2x_1 - 3x_2 + x_3 &\leqslant 9 \\ x_1, x_2, x_3 &\geqslant 0 \end{aligned}$$

Starting from the all-slack point, perform two iterations of the simplex method. Write down the basic feasible point arrived at by the end of the second iteration.

End-of-section Exercises

Exercise 3.6

You obtained the canonical form of the model for the smallholder problem (Exercise 1.7) in Exercise 1.13 as:

maximize $z = 30x_1 + 240x_2 + 500x_3$

subject to

$$\begin{aligned}
0.01x_1 + 0.2x_2 + x_3 + x_4 &= 25 && \text{(land)}\\
20x_1 + 120x_2 + 240x_3 + x_5 &= 4000 && \text{(money)}\\
20x_2 + 30x_3 + x_6 &= 1000 && \text{(labour)}\\
x_1 + x_7 &= 200 && \text{(hens)}
\end{aligned}$$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geqslant 0$

Solve this problem by the simplex method, and interpret the solution.

4 Matrix formulation of the method

In this section, we look at the simplex method in more detail. In Subsection 4.4 we discuss its theoretical justification and consider some difficulties that may arise, but first, in Subsections 4.1–4.3, we formalize the algebraic description of the simplex method by using matrices. To do this we shall need the matrix representation of the canonical form of a linear programming model, i.e.

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{A}$ is an $m \times (n+m)$ matrix and $\mathbf{b} \geqslant \mathbf{0}$. We shall assume throughout Subsections 4.1–4.3 that there is no degeneracy, so that exactly n of the variables are zero at each basic feasible point; we consider degeneracy in Subsection 4.4.

4.1 Determination of a basic feasible point

Step (a) of the simplex method involves determining a basic feasible point. In order to describe this process in terms of matrices, we need to divide the model into its basic and non-basic parts.

Example 4.1

Express step (a) of the first iteration of the simplex method in matrix notation, for the following linear programming model:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [6, 4, 1, 0, 0, 0]^T,\ \mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T,$

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 2 & 1 & 0 & 0 \\ 2 & 1 & -1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 8 \\ 6 \\ 5 \end{bmatrix}.$$

Solution

This model is already in canonical form. For step (a) of the first iteration of the simplex method, we want to find a basic feasible point, choosing the all-slack point if this is feasible. To find the all-slack point, we set $x_1 = x_2 = x_3 = 0$ and find values for x_4, x_5 and x_6 by solving

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 8 \\ 6 \\ 5 \end{bmatrix}.$$

We then substitute these values of the variables into the objective function to find the corresponding value of z.

This procedure can be written more succinctly using matrix notation. To do this we first reorder and partition $\mathbf{x}$ so that the basic variables come first, writing $\mathbf{x} = [\mathbf{x}_\mathrm{B}^T | \mathbf{x}_\mathrm{N}^T]^T$ where $\mathbf{x}_\mathrm{B} = [x_4, x_5, x_6]^T$ and $\mathbf{x}_\mathrm{N} = [x_1, x_2, x_3]^T$. Similarly, we reorder and partition the matrix $\mathbf{A}$ into sets of columns corresponding to the basic and non-basic variables:

Note that within a computer package there would be no need to reorder the vectors or the columns of the coefficient matrix, as a record would be kept of the basic variables and the columns which form the basis matrix. We do so here for clarity.

$$\mathbf{A} = [\mathbf{B}|\mathbf{N}] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & -1 & 2 \\ 0 & 1 & 0 & 2 & 1 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right].$$

The square matrix $\mathbf{B}$ is called the **basis matrix**. The vector $\mathbf{c}$ can also be reordered and partitioned into $\mathbf{c}_\mathrm{B}$ and $\mathbf{c}_\mathrm{N}$: $\mathbf{c} = [\mathbf{c}_\mathrm{B}^T | \mathbf{c}_\mathrm{N}^T]^T = [0, 0, 0 | 6, 4, 1]^T$.

The objective function can thus be written as

$$z = \mathbf{c}^T\mathbf{x} = \mathbf{c}_\mathrm{B}^T\mathbf{x}_\mathrm{B} + \mathbf{c}_\mathrm{N}^T\mathbf{x}_\mathrm{N}$$

and the constraints as

$$\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}_\mathrm{B} + \mathbf{N}\mathbf{x}_\mathrm{N} = \mathbf{b},$$
$$\mathbf{x}_\mathrm{B} \geqslant \mathbf{0}, \mathbf{x}_\mathrm{N} \geqslant \mathbf{0}.$$

Thus, to find the all-slack point we set $\mathbf{x}_\mathrm{N} = \mathbf{0}$ and solve $\mathbf{B}\mathbf{x}_\mathrm{B} = \mathbf{b}$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_\mathrm{B} = \begin{bmatrix} 8 \\ 6 \\ 5 \end{bmatrix}.$$

This gives $x_{\mathrm{B}1}(= x_4) = 8$, $x_{\mathrm{B}2}(= x_5) = 6$ and $x_{\mathrm{B}3}(= x_6) = 5$. Since $\mathbf{x}_\mathrm{B} \geqslant \mathbf{0}$, this point is feasible. We can then calculate the corresponding value of the objective function using $z = \mathbf{c}_\mathrm{B}^T\mathbf{x}_\mathrm{B} + \mathbf{c}_\mathrm{N}^T\mathbf{x}_\mathrm{N} = \mathbf{c}_\mathrm{B}^T\mathbf{x}_\mathrm{B}$ (since $\mathbf{x}_\mathrm{N} = \mathbf{0}$), where $\mathbf{c}_\mathrm{B}^T = [0, 0, 0]$ and $\mathbf{x}_\mathrm{B} = [8, 6, 5]^T$. This gives $z = 0$, as expected. ■

Recall, from Section 2, that a feasible point is one that satisfies not only $\mathbf{A}\mathbf{x} = \mathbf{b}$ but also $\mathbf{x} \geqslant \mathbf{0}$, so we need to check that $\mathbf{x}_\mathrm{B} \geqslant \mathbf{0}$.

The canonical form of a general linear programming model, given at the start of this section, can be partitioned in the same way. It has m constraint equations and $n + m$ variables. Let $\mathbf{x}_\mathrm{B}$ be the vector of basic variables, and $\mathbf{c}_\mathrm{B}$ the cost vector corresponding to $\mathbf{x}_\mathrm{B}$; then $\mathbf{x}_\mathrm{B}$ and $\mathbf{c}_\mathrm{B}$ will each have m elements. Let the current basis list B be the set $\{j : x_j \in \mathbf{x}_\mathrm{B}\}$. Let $\mathbf{B}$ be the basis matrix consisting of the columns of $\mathbf{A}$ corresponding to the basis list B; the basis matrix $\mathbf{B}$ will be an $m \times m$ matrix. The n non-basic variables then form the vector $\mathbf{x}_\mathrm{N}$, the corresponding cost vector is $\mathbf{c}_\mathrm{N}$ and the remaining n columns of $\mathbf{A}$ form the $m \times n$ matrix $\mathbf{N}$. The canonical form of the model then becomes:

$$\text{maximize } z = \mathbf{c}_\mathrm{B}^T\mathbf{x}_\mathrm{B} + \mathbf{c}_\mathrm{N}^T\mathbf{x}_\mathrm{N}$$
$$\text{subject to } \mathbf{B}\mathbf{x}_\mathrm{B} + \mathbf{N}\mathbf{x}_\mathrm{N} = \mathbf{b},\ \mathbf{x}_\mathrm{B} \geqslant \mathbf{0},\ \mathbf{x}_\mathrm{N} \geqslant \mathbf{0}$$

To find the current basic feasible point, we therefore set $\mathbf{x}_N = \mathbf{0}$, solve $\mathbf{B}\mathbf{x}_B = \mathbf{b}$ to give ${}_{\text{old}}\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$, and evaluate ${}_{\text{old}}z = \mathbf{c}_{B\,\text{old}}^T\mathbf{x}_B$.

We use the notation ${}_{\text{old}}\mathbf{x}_B$ and ${}_{\text{old}}z$ to distinguish these values, calculated in step (a) of the simplex method, from the new values of $\mathbf{x}_B$ and z at the basic feasible point we are about to find in steps (b) and (c).

Exercise 4.1

The canonical form of the model for the gardening problem is

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [3, 5, 4, 0, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 600 \\ 400 \\ 300 \\ 80 \end{bmatrix}.$$

Write down the vectors $\mathbf{x}_B$, $\mathbf{x}_N$, $\mathbf{c}_B$, $\mathbf{c}_N$ and the matrices $\mathbf{B}$ and $\mathbf{N}$ for the all-slack point, and use these to calculate ${}_{\text{old}}\mathbf{x}_B$ and ${}_{\text{old}}z$.

At this stage you may be wondering why we bother with the non-basic components of the model — since $\mathbf{x}_N = \mathbf{0}$ it is tempting to ignore $\mathbf{N}$ and $\mathbf{c}_N$. However, remember that step (b) of the simplex method involves expressing the basic variables and the objective function in terms of the non-basic variables in order to investigate the effects of moving away from the current basic feasible point, for which we shall need $\mathbf{N}$ and $\mathbf{c}_N$.

4.2 Choice of the new basic variable

We now have a current basic feasible point, ${}_{\text{old}}\mathbf{x}_B$, and want to see what happens to the value of the objective function as we move away from it. The first thing we need to do is to express the basic variables in terms of the non-basic ones, so that we can then express z in terms of the non-basic variables.

Example 4.2

For the all-slack point of the model in Example 4.1, express the basic variables in terms of the non-basic ones using matrix notation.

Solution

A basic feasible point must satisfy the constraints $\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$, so

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N.$$

For the all-slack point of Example 4.1 we therefore have, since $\mathbf{B}^{-1} = \mathbf{B} = \mathbf{I}$,

$$\begin{matrix} \mathbf{x}_B \\ \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} \end{matrix} = \begin{matrix} \mathbf{B}^{-1}\mathbf{b} \\ \begin{bmatrix} 8 \\ 6 \\ 5 \end{bmatrix} \end{matrix} - \begin{matrix} \mathbf{B}^{-1}\mathbf{N} \\ \begin{bmatrix} 1 & -1 & 2 \\ 2 & 1 & -1 \\ 1 & 1 & 0 \end{bmatrix} \end{matrix} \begin{matrix} \mathbf{x}_N \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \end{matrix},$$

so we have expressed the basic variables x_4, x_5 and x_6 in terms of the non-basic ones x_1, x_2 and x_3 as required. ■

For any basic feasible point we have $\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}$, where $\mathbf{x}_B \geqslant \mathbf{0}$ and $\mathbf{x}_N \geqslant \mathbf{0}$. Thus, rearranging, we have

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N = {}_{\text{old}}\mathbf{x}_B - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N, \tag{4.1}$$

which expresses the basic variables in terms of the non-basic ones.

Remember that, by definition, ${}_{\text{old}}\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$.

Now, to see which non-basic variable, when increased in value, will cause the most rapid increase in the value of the objective function, we need to express z in terms of the non-basic variables. We have

$$\begin{aligned} z &= \mathbf{c}_{\mathrm{B}}^T\mathbf{x}_{\mathrm{B}} + \mathbf{c}_{\mathrm{N}}^T\mathbf{x}_{\mathrm{N}} \\ &= \mathbf{c}_{\mathrm{B}}^T({}_{\mathrm{old}}\mathbf{x}_{\mathrm{B}} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathrm{N}}) + \mathbf{c}_{\mathrm{N}}^T\mathbf{x}_{\mathrm{N}} \qquad \text{(by Equation (4.1))} \\ &= \mathbf{c}_{B\,\mathrm{old}}^T\mathbf{x}_{\mathrm{B}} - (\mathbf{c}_{\mathrm{B}}^T\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_{\mathrm{N}}^T)\mathbf{x}_{\mathrm{N}} \\ &= {}_{\mathrm{old}}z - \sum_{j\notin B}(\mathbf{w}^T\mathbf{a}_j - c_j)x_j, \end{aligned}$$

where B is the current basis list, $\mathbf{w}^T = \mathbf{c}_{\mathrm{B}}^T\mathbf{B}^{-1}$ and $\mathbf{a}_j$ is the column of $\mathbf{N}$ (and hence of $\mathbf{A}$) corresponding to the non-basic variable x_j. The coefficients $(\mathbf{w}^T\mathbf{a}_j - c_j)$ are the reduced costs, d_j, and we find the most negative of these to determine the non-basic variable x_q that becomes basic, i.e. to determine the non-basic variable x_q which, when increased in value, causes the greatest rate of increase in the value of z. If no d_j is negative then no increase in any of the non-basic variables will improve the value of the objective function, and so an optimal vertex has been reached.

Example 4.3

Using matrices, determine the first variable to become basic for the model in Example 4.1.

Solution

For the all-slack point it is easy to choose the variable to become basic by inspection. We have

$$z = \mathbf{c}_{\mathrm{B}}^T\mathbf{x}_{\mathrm{B}} + \mathbf{c}_{\mathrm{N}}^T\mathbf{x}_{\mathrm{N}} = 0x_4 + 0x_5 + 0x_6 + 6x_1 + 4x_2 + x_3.$$

Clearly x_1 is the variable that, when increased in value, will cause the greatest rate of increase in the value of z. Using matrices, we have:

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B}^{-1}\mathbf{N} = \mathbf{N} = \begin{bmatrix} 1 & -1 & 2 \\ 2 & 1 & -1 \\ 1 & 1 & 0 \end{bmatrix};$$

$$\mathbf{c}_{\mathrm{B}}^T = [0,0,0], \quad \mathbf{c}_{\mathrm{N}}^T = [6,4,1], \quad \mathbf{w}^T = \mathbf{c}_{\mathrm{B}}^T\mathbf{B}^{-1} = [0,0,0], \quad B = \boxed{4}\boxed{5}\boxed{6}.$$

Therefore

$$d_1 = \mathbf{w}^T\mathbf{a}_1 - c_1 = [0,0,0]\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} - 6 = -6,$$

$$d_2 = \mathbf{w}^T\mathbf{a}_2 - c_2 = [0,0,0]\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} - 4 = -4,$$

$$d_3 = \mathbf{w}^T\mathbf{a}_3 - c_3 = [0,0,0]\begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} - 1 = -1.$$

The most negative reduced cost is d_1, and so x_1 will become basic first. ■

Exercise 4.2

Use your answer to Exercise 4.1 to express the objective function for the gardening problem in terms of the non-basic variables x_1, x_2 and x_3, and hence determine which of these variables should become basic.

Note that the calculation of the reduced costs can be expressed in vector form as

$$\mathbf{d}^T = \mathbf{w}^T\mathbf{N} - \mathbf{c}_{\mathrm{N}}^T.$$

The vector $\mathbf{d}$ is referred to as the **reduced cost vector**.

4.3 Determination of the new active constraint

Expressing $\mathbf{x}_{\mathrm{B}}$ in terms of $\mathbf{x}_{\mathrm{N}}$ also enables us to determine which variable becomes zero first, and hence which variable becomes non-basic and which constraint becomes active. We have, from Equation (4.1), that

$$\mathbf{x}_{\mathrm{B}} = {}_{\mathrm{old}}\mathbf{x}_{\mathrm{B}} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_{\mathrm{N}} = {}_{\mathrm{old}}\mathbf{x}_{\mathrm{B}} - \sum_{j \notin B} \mathbf{y}_j x_j, \quad \text{where } \mathbf{y}_j = \mathbf{B}^{-1}\mathbf{a}_j.$$

If x_q becomes basic and takes the value θ, where θ is increased from zero while the other non-basic variables remain zero, this becomes

$$\mathbf{x}_{\mathrm{B}} = {}_{\mathrm{old}}\mathbf{x}_{\mathrm{B}} - \mathbf{y}_q\theta.$$

To maintain feasibility we must have $\mathbf{x}_{\mathrm{B}} \geqslant \mathbf{0}$, so we can only increase θ until one of the basic variables becomes zero. We can find out which variable becomes zero first by setting each element $x_{\mathrm{B}k}$ of $\mathbf{x}_{\mathrm{B}}$ to zero in turn and solving for θ. Thus, writing θ_k for the value of θ corresponding to $x_{\mathrm{B}k} = 0$, we have

$$0 = {}_{\mathrm{old}}x_{\mathrm{B}k} - (y_q)_k\,\theta_k,$$

where ${}_{\mathrm{old}}x_{\mathrm{B}k}$ is the kth element of ${}_{\mathrm{old}}\mathbf{x}_{\mathrm{B}}$ and $(y_q)_k$ is the kth element of $\mathbf{y}_q$, giving

$$\theta_k = \frac{{}_{\mathrm{old}}x_{\mathrm{B}k}}{(y_q)_k}.$$

We want the variable x_p corresponding to the least non-negative value of θ_k, i.e. if

$$\theta_s = \min_{\theta_k \geqslant 0} \theta_k$$

then we want the variable $x_p = x_{\mathrm{B}s}$, corresponding to θ_s, to become non-basic, and the constraint corresponding to $x_p = 0$ to become active. If $\theta_k < 0$ for all k then the model is unbounded.

Comparing the values of θ_k to find the least non-negative one is often referred to as the **ratio test**.

Example 4.4

Determine the first constraint to become active for the model in Example 4.1.

Solution

As we saw in Example 4.3, the variable x_1 has become basic, and the current basis list is $\boxed{4 \mid 5 \mid 6}$. We also have ${}_{\mathrm{old}}\mathbf{x}_{\mathrm{B}} = [8, 6, 5]^T$ and $\mathbf{y}_1 = [1, 2, 1]^T$. The ratio test therefore gives $\theta_1 = \frac{8}{1} = 8$, $\theta_2 = \frac{6}{2} = 3$ and $\theta_3 = \frac{5}{1} = 5$. All the ratios are positive and the smallest, θ_2, corresponds to $x_{\mathrm{B}2} = x_5$. So x_5 becomes non-basic first, and the constraint corresponding to $x_5 = 0$ becomes active. ■

Exercise 4.3

Use your answers to Exercises 4.1 and 4.2, and the matrix theory above, to determine which constraint becomes active in the first iteration of the simplex method for the gardening problem.

Procedure 4.1 Simplex method for solving linear programming models (matrix version)

Given a basis list B corresponding to a basic feasible point of the linear programming model given in canonical form by

$$\text{maximize } z = \mathbf{c}^T\mathbf{x} = \mathbf{c}_{\mathrm{B}}^T\mathbf{x}_{\mathrm{B}} + \mathbf{c}_{\mathrm{N}}^T\mathbf{x}_{\mathrm{N}}$$
$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}_{\mathrm{B}} + \mathbf{N}\mathbf{x}_{\mathrm{N}} = \mathbf{b},\ \mathbf{x} = [\mathbf{x}_{\mathrm{B}}^T|\mathbf{x}_{\mathrm{N}}^T]^T \geqslant \mathbf{0}$$

where the elements of $\mathbf{x}_{\mathrm{B}}$ are the basic variables and the elements of $\mathbf{x}_{\mathrm{N}}$ are the non-basic variables, to solve the model by the simplex method, repeat the following steps.

(a) *Determine the current basic feasible point*

Find the current basic feasible point by setting $\mathbf{x}_{\mathrm{N}} = \mathbf{0}$ and solving $\mathbf{B}\mathbf{x}_{\mathrm{B}} = \mathbf{b}$ to give

$$_{\mathrm{old}}\mathbf{x}_{\mathrm{B}} = \mathbf{B}^{-1}\mathbf{b}.$$

Evaluate the objective function at the current point using

$$_{\mathrm{old}}z = \mathbf{c}_{\mathrm{B}\,\mathrm{old}}^T\mathbf{x}_{\mathrm{B}}.$$

(b) *Choose the variable to become basic*

Determine the reduced cost vector

$$\mathbf{d}^T = \mathbf{w}^T\mathbf{N} - \mathbf{c}_{\mathrm{N}}^T,$$

where $\mathbf{w}^T = \mathbf{c}_{\mathrm{B}}^T\mathbf{B}^{-1}$. If all the reduced costs are non-negative, stop, as a solution has been reached (the optimizer is $_{\mathrm{old}}\mathbf{x}_{\mathrm{B}}$ and the optimal value of the objective function is $_{\mathrm{old}}z$). Otherwise, choose the variable x_q corresponding to the most negative reduced cost to become basic.

(c) *Determine the constraint to become active*

Calculate

$$\mathbf{y}_q = \mathbf{B}^{-1}\mathbf{a}_q,$$

where $\mathbf{a}_q$ is the column of $\mathbf{N}$ corresponding to x_q, and use the ratio test to determine

$$\theta_s = \min_{\theta_k \geqslant 0} \theta_k,$$

where $\theta_k = {}_{\mathrm{old}}x_{\mathrm{B}k}/(y_q)_k$. If every θ_k is negative, stop, as the problem is unbounded. Otherwise, the variable to become non-basic is the variable $x_p = x_{\mathrm{B}s}$ in $\mathbf{x}_{\mathrm{B}}$ corresponding to θ_s, and the corresponding constraint becomes active. Update the basis list, replacing p by q.

Note that the simplex method requires a new value for $\mathbf{B}^{-1}$ at each iteration. For large problems, the calculation of $\mathbf{B}^{-1}$ is in general very time consuming and so, as *Unit I.2* advises, we might consider not calculating $\mathbf{B}^{-1}$ but rather solving $\mathbf{B}\mathbf{x}_{\mathrm{B}} = \mathbf{b}$, $\mathbf{w}^T\mathbf{B} = \mathbf{c}_{\mathrm{B}}^T$ and $\mathbf{B}\mathbf{y}_q = \mathbf{a}_q$ for $\mathbf{x}_{\mathrm{B}}$, $\mathbf{w}$ and $\mathbf{y}_q$ using the $\mathbf{LU}$ decomposition of $\mathbf{B}$. However, as only one column of $\mathbf{B}$ changes at each iteration of the simplex method, there is a relatively simple and efficient technique for updating $\mathbf{B}^{-1}$ from one iteration to the next, the details of which we omit, which is generally used instead. Thus $\mathbf{B}^{-1}$ needs to be calculated from scratch just once, at the start of the method, at which point $\mathbf{B}$ usually takes a very simple form from which $\mathbf{B}^{-1}$ is easily computed.

Exercise 4.4

Starting from the all-slack point, perform two iterations of the matrix version of the simplex method on the model of Exercise 3.5, which in canonical form is given by

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$$

where $\mathbf{c} = [2, -5, 3, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 3 & -2 & 1 & 0 & 1 & 0 \\ 2 & -3 & 1 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 12 \\ 8 \\ 9 \end{bmatrix}.$$

(*Hint*: Note that

$$\begin{bmatrix} 3 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & 0 & 1 \end{bmatrix}.)$$

4.4 Some background theory

So far we have assumed that the simplex method can be applied as long as we can find a basic feasible point to start with. We have also assumed that, provided the model is bounded, the method will lead to the solution and that it will do so in a finite number of steps. We now justify these assumptions and describe the difficulties that may arise. We shall continue to refer to the canonical form of a linear programming model, as outlined at the start of this section.

For the constraint equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ to have at least one basic feasible point, the m rows of the $m \times (n + m)$ coefficient matrix $\mathbf{A}$ must be linearly independent. That is, the coefficient matrix $\mathbf{A}$ must have rank m, i.e. be of full rank.

The **rank** of a matrix is its number of linearly independent rows. If all its rows are linearly independent, it is said to have **full rank**.

If the m rows of $\mathbf{A}$ are *not* linearly independent, i.e. if the matrix is not of full rank, one of the following must hold.

- The constraint equations are *contradictory*, in which case the model has no solution.
- There is *redundancy* among the constraint equations, in that some can be expressed as linear combinations of the others. Such redundancy should be sought out and eliminated before solution is attempted. This means removing some of the constraint equations, reducing m until the matrix is of full rank.

Redundancy among the *constraint equations* should not be confused with redundancy among the *constraints*. Where slack variables have been introduced, the constraint equations must be linearly independent because such variables are unique to a particular equation. Thus only those constraint equations based on equality constraints in the general form of the model can be redundant in this context.

The condition that the coefficient matrix $\mathbf{A}$ should be of full rank leads to the algebraic interpretation of Theorem 2.2 (Optimal Vertex Theorem). This new theorem, whose proof we omit, is the key to the solubility of linear programming models.

Theorem 4.1 Fundamental Theorem of Linear Programming

Given a linear programming model in canonical form where the coefficient matrix of the constraint equations is of full rank:

(a) if there is a feasible point, then there is a basic feasible point;

(b) if there is a solution, then there is a solution at a basic feasible point.

This theorem means that, provided a linear programming model is feasible, i.e. its feasible region is non-empty, then there must exist at least one basic feasible point for which we can define a basis list and which we can use as a starting point for the simplex method.

We shall see how to find basic feasible points in *Unit II.2.*

However, Theorem 4.1 does not guarantee that the simplex method will always either give us a solution or tell us that the problem is unbounded in a finite number of iterations. To ensure this we need to investigate the properties of basic feasible points. First, we need to revise our definitions of basis list, basic variable and non-basic variable, to take account of the possibility of degeneracy.

Definitions

For a linear programming model in canonical form with m linearly independent constraint equations and $n + m$ variables numbered $1, 2, \ldots, n + m$, a **basis list** is any set of m of these numbers. The **basic variables** are those whose numbers are in the basis list; the remaining variables are the **non-basic variables**.

Notice that this definition makes no mention of non-zero values for the basic variables or zero values for the non-basic ones. We can associate a basis list with each vertex of the model, in such a way that each non-basic variable is zero at the vertex; however, this does not restrict the basic variables to non-zero values at the vertex. We are now allowing some of them to be zero, in which case more than n constraint bounds would intersect at the vertex. If feasible, this would therefore be a degenerate vertex. Thus an alternative definition of a **degenerate vertex**, which we shall also refer to as a **degenerate basic feasible point**, in that it is a feasible vertex $\mathbf{x} = [\mathbf{x}_{\mathrm{B}}^T | \mathbf{x}_{\mathrm{N}}^T]^T$ for which $\mathbf{x}_{\mathrm{B}k} = 0$ for some k.

The problem with degenerate basic feasible points is that, in theory, they may cause the simplex iterations to cycle round a sequence of vertices without improving the value of the objective function. If some $x_{\mathrm{B}k} = 0$, then it could be chosen as the variable to become non-basic, but there would be no change in the value of the objective function. Thus, to ensure that the simplex method terminates in a finite number of iterations we need to ensure that none of the basic feasible points is degenerate.

This leads us to the following theorem, whose proof we omit.

Theorem 4.2 Simplex Method Termination Theorem

If the coefficient matrix of the constraint equations of the canonical form of a linear programming model is of full rank and none of the model's basic feasible points is degenerate, then the simplex method applied to that model terminates in a finite number of iterations.

We should emphasize that the theorem does not state that a solution will be found. It merely states that the simplex method will terminate, and termination can occur not only when a solution is found, but also when unboundedness is detected. However, *if* a solution exists (i.e. if the model is feasible and bounded), then the theorem does tell us that the simplex method will find it in a finite number of iterations, provided the conditions of Theorem 4.2 are satisfied.

The simplex method may still work for a model that suffers from degeneracy. The method can also be adapted to deal with degeneracy, as we shall see in *Unit II.2.*

Exercise 4.5

Given that the model for the Christmas hamper problem (Exercise 1.14) does not suffer from degeneracy, explain why you would expect the simplex method to find a solution to this problem.

4.5 Computer activities

Activity 4.1

Use the simplex method to solve:

(a) the model in Exercise 3.5;

(b) the model in Example 4.1.

Activity 4.2

Use the simplex method to solve the table manufacturing problem (Example 1.1). Compare your solution with that obtained in Example 2.3.

Activity 4.3

Use the simplex method to solve:

(a) the smallholder problem (Exercise 1.7);

(b) the gardening problem (Exercise 1.15).

Compare your solutions with those obtained in Exercise 3.6 and Example 3.1 respectively.

Activity 4.4

Use the simplex method to solve:

(a) the Christmas hamper problem (Exercise 1.14);

(b) the margarine blending problem (Example 1.2).

Interpret your solutions.

Solutions to the exercises

Section 1

1.1 We follow Procedure 1.1.

(a) The objective is to minimize the daily cost of feeding ten pigs. (It would be equally correct to define the objective as minimizing the daily cost of feeding one pig.) Since the costs are given in pounds per tonne and as each pig's daily requirement is measured in kilograms, with just ten pigs it would be sensible to measure the objective function in pence.

(b) The only variables are the amounts of fish meal and meat scraps fed to the pigs each day. Both variables can be measured in kilograms.

(c) The constraints are the pigs' minimum daily requirements of protein and amino acids and their maximum daily amount of calcium.

The parameters are the amounts of protein, calcium and amino acids in the fish meal and meat scraps, the limits on the pigs' daily intake of protein, calcium and amino acids, and the cost of fish meal and meat scraps.

(d) We shall use the following definitions:

z the daily cost (in pence) of feeding the ten pigs;
x_1 the daily amount (in kg) of fish meal fed to the ten pigs;
x_2 the daily amount (in kg) of meat scraps fed to the ten pigs.

(e)

	fish meal	meat scraps	limit (for ten pigs)
cost (pence per kg)	15	12	—
protein (kg per kg)	0.6	0.5	$\geqslant 16$
calcium (kg per kg)	0.05	0.11	$\leqslant 3$
amino acids (kg per kg)	0.18	0.05	$\geqslant 3$

(f) The objective is

$$\text{minimize } z = 15x_1 + 12x_2.$$

(g) The non-trivial constraints are

$$\begin{aligned} 0.6x_1 + 0.5x_2 &\geqslant 16 \quad \text{(protein)},\\ 0.05x_1 + 0.11x_2 &\leqslant 3 \quad \text{(calcium)},\\ 0.18x_1 + 0.05x_2 &\geqslant 3 \quad \text{(amino acids)}. \end{aligned}$$

(h) As the farmer cannot feed a negative amount to the pigs, we have the trivial constraints $x_1 \geqslant 0$ and $x_2 \geqslant 0$.

The complete model is

$$\begin{aligned} &\text{minimize } z = 15x_1 + 12x_2\\ &\text{subject to}\\ &0.6x_1 + 0.5x_2 \geqslant 16 \quad \text{(protein)}\\ &0.05x_1 + 0.11x_2 \leqslant 3 \quad \text{(calcium)}\\ &0.18x_1 + 0.05x_2 \geqslant 3 \quad \text{(amino acids)}\\ &x_1, x_2 \geqslant 0 \end{aligned}$$

(If you chose to scale the objective function and variables differently, your model will contain different numerical values. For example, you may have chosen to minimize the daily cost of feeding one pig, or you may have chosen to measure z in pounds. Your choice of scaling does not matter, as long as you were consistent and as long as your model is reasonably well scaled.)

1.2 The variables could be measured in tonnes of oil per month, but, since the refining restrictions are given as 800 and 1000 tonnes per month, it would seem sensible to work in hundreds of tonnes per month, both for the oils and for the margarine.

1.3 Four obvious constraints are:

- a limit of 1000 tonnes per month on the sum of the quantities of ground-nut, soya-bean and palm oil refined;
- a limit of 800 tonnes per month on the sum of the quantities of lard and fish oil refined;
- a lower hardness constraint of 5.6 on the margarine;
- an upper hardness constraint of 7.4 on the margarine.

There is one further constraint: since we have included the total quantity of margarine produced as a variable, we must also include the *quantity constraint* that the amount of margarine produced must be the sum of the amounts of oil used. (These two amounts must be equal as we are told that the amount of oil lost during the refining and blending processes is negligible.)

The parameters are the hardnesses of the oils, the limits on the hardness of the margarine, the limits on the refining capacities and the costs of the oils.

1.4 The objective function, z, is the income from selling the margarine less the costs of the oils, i.e.

$$z = -9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5 + 24x_6,$$

and the objective is to maximize this.

1.5 The vegetable refining capacity constraint (in units of hundreds of tonnes) is

$$x_1 + x_2 + x_3 \leqslant 10.$$

The non-vegetable refining capacity constraint (in units of hundreds of tonnes) is

$$x_4 + x_5 \leqslant 8.$$

1.6 The upper hardness constraint is

$$1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 7.4x_6 \leqslant 0.$$

1.7 We follow Procedure 1.1.

(a) The objective is to maximize annual income, which is measured in pounds. (You may have chosen units of tens, hundreds or even thousands of pounds.)

(b) The variables are the numbers of hens, pigs and bullocks kept.

(c) The constraints are the availability of land, the amount available to spend on food, the number of hens that can be managed, and the number of hours available for feeding the bullocks and pigs per year.

The parameters are the land requirements, food costs and labour requirements for each type of animal, the limits on the available land, money and labour, and the annual income for each type of animal.

(d) We shall use the following definitions:

z the annual income, in pounds;
x_1 the number of hens;
x_2 the number of pigs;
x_3 the number of bullocks.

(e) The following table shows the relationships between the variables and the objective function and between the variables and the constraints.

	each hen	each pig	each bullock	upper limit
income (£ per year)	30	240	500	—
land requirements (hectares)	0.01	0.2	1	25
food costs (£ per year)	20	120	240	4000
labour requirements (hours per year)	0	20	30	1000
hens (number per year)	1	0	0	200

(f) The objective is

$$\text{maximize } z = 30x_1 + 240x_2 + 500x_3.$$

(g) The non-trivial constraints are:

$$\begin{aligned} 0.01x_1 + 0.2x_2 + x_3 &\leqslant 25 && \text{(land)}\\ 20x_1 + 120x_2 + 240x_3 &\leqslant 4000 && \text{(money)}\\ 20x_2 + 30x_3 &\leqslant 1000 && \text{(labour)}\\ x_1 &\leqslant 200 && \text{(hens)} \end{aligned}$$

(h) The trivial constraints are

$$x_1, x_2, x_3 \geqslant 0.$$

The complete model is:

$$\text{maximize } z = 30x_1 + 240x_2 + 500x_3$$

subject to

$$\begin{aligned} 0.01x_1 + 0.2x_2 + x_3 &\leqslant 25 && \text{(land)}\\ 20x_1 + 120x_2 + 240x_3 &\leqslant 4000 && \text{(money)}\\ 20x_2 + 30x_3 &\leqslant 1000 && \text{(labour)}\\ x_1 &\leqslant 200 && \text{(hens)}\\ x_1, x_2, x_3 &\geqslant 0 \end{aligned}$$

(A case could be made for some scaling of the variables and parameters, though we shall not pursue this here.)

1.8 The non-negativity constraints can be expressed as $\mathbf{x} \geqslant \mathbf{0}$, where $\mathbf{0}$ is the zero vector.

1.9 **(a)** The variable x_6 must be replaced by $x_1 + x_2 + x_3 + x_4 + x_5$ whenever it appears.

The objective function becomes

$$\begin{aligned} z = &-9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5\\ &+ 24(x_1 + x_2 + x_3 + x_4 + x_5)\\ &= 14.4x_1 + 8x_2 + 8x_3 + 2x_4 + 5x_5. \end{aligned}$$

The vegetable and non-vegetable refining capacity constraints are already in the required form.

The lower hardness constraint contains x_6, so it must be changed to

$$\begin{aligned} &1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5\\ &\quad - 5.6(x_1 + x_2 + x_3 + x_4 + x_5) \geqslant 0, \end{aligned}$$

i.e. to

$$-4.4x_1 - 2.2x_2 + 2.4x_3 + 5.2x_4 + 2.7x_5 \geqslant 0.$$

The signs must also be changed to make it $\leqslant$:

$$4.4x_1 + 2.2x_2 - 2.4x_3 - 5.2x_4 - 2.7x_5 \leqslant 0.$$

Replacing x_6 in the upper hardness constraint gives

$$-6.2x_1 - 4.0x_2 + 0.6x_3 + 3.4x_4 + 0.9x_5 \leqslant 0.$$

The quantity restraint is no longer required.

The non-negativity constraint $x_6 \geqslant 0$ must be replaced by the constraint.

$$x_1 + x_2 + x_3 + x_4 + x_5 \geqslant 0.$$

However, their sum must be non-negative since $x_1, x_2, x_3, x_4, x_5 \geqslant 0$, so this last constraint can be omitted.

The model is now

$$\text{maximize } z = 14.4x_1 + 8x_2 + 8x_3 + 2x_4 + 5x_5$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &\leqslant 10\\ x_4 + x_5 &\leqslant 8\\ 4.4x_1 + 2.2x_2 - 2.4x_3 - 5.2x_4 - 2.7x_5 &\leqslant 0\\ -6.2x_1 - 4.0x_2 + 0.6x_3 + 3.4x_4 + 0.9x_5 &\leqslant 0\\ x_1, x_2, x_3, x_4, x_5 &\geqslant 0 \end{aligned}$$

In matrix notation, this is

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$

subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [14.4, 8, 8, 2, 5]^T$, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 1\\ 4.4 & 2.2 & -2.4 & -5.2 & -2.7\\ -6.2 & -4.0 & 0.6 & 3.4 & 0.9 \end{bmatrix},$$

$\mathbf{b} = [10, 8, 0, 0]^T$.

(b) This time, x_6 remains in the model, so the objective function is the same as in the original model. The refining capacity and upper hardness constraints are also unchanged. The sign of the lower hardness constraint must be changed to give

$$-1.2x_1 - 3.4x_2 - 8.0x_3 - 10.8x_4 - 8.3x_5 + 5.6x_6 \leqslant 0.$$

Finally, the quantity constraint must be replaced by the two constraints

$$x_1 + x_2 + x_3 + x_4 + x_5 - x_6 \leqslant 0$$
$$-x_1 - x_2 - x_3 - x_4 - x_5 + x_6 \leqslant 0$$

The model thus becomes

maximize

$$z = -9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5 + 24x_6$$

subject to

$$\begin{aligned}
x_1 + x_2 + x_3 &\leqslant 10\\
x_4 + x_5 &\leqslant 8\\
-1.2x_1 - 3.4x_2 - 8.0x_3 - 10.8x_4 - 8.3x_5 + 5.6x_6 &\leqslant 0\\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 7.4x_6 &\leqslant 0\\
x_1 + x_2 + x_3 + x_4 + x_5 - x_6 &\leqslant 0\\
-x_1 - x_2 - x_3 - x_4 - x_5 + x_6 &\leqslant 0\\
x_1, x_2, x_3, x_4, x_5, x_6 &\geqslant 0
\end{aligned}$$

In matrix notation this is

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [-9.6, -16, -16, -22, 19, 24]^T$,
$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T$, $\mathbf{b} = [10, 8, 0, 0, 0, 0]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ -1.2 & -3.4 & -8.0 & -10.8 & -8.3 & 5.6 \\ 1.2 & 3.4 & 8.0 & 10.8 & 8.3 & -7.4 \\ 1 & 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

(Notice that just changing the way the equality constraint is treated has created different models in standard form. However, as these different models represent the same problem, they should both give rise to the same answer to the original problem.)

1.10 We follow Procedure 1.2.

(a) As there is a constant term in the objective function, we replace $z'' = x_1 + 2x_2 + 6$ by $z' = x_1 + 2x_2$ to give

minimize $z' = z_1 + 2x_2$,

where $z' = z'' - 6$.

(b) The problem is stated as a minimization, so we need to rewrite the objective as

maximize $z = -x_1 - 2x_2$,

where $z = -z'$.

(c) There is no constraint $x_2 \geqslant 0$, so x_2 appears to be a free variable. We must replace x_2 by $x_2 - x_5$, where $x_2 \geqslant 0$ and $x_5 \geqslant 0$. The objective then becomes

maximize $z = -x_1 - 2x_2 + 2x_5$,

and the first and third constraints become

$$\begin{aligned}
x_1 - x_2 - x_3 + x_5 &\leqslant -7\\
x_2 - x_3 + 2x_4 - x_5 &\geqslant 0
\end{aligned}$$

(d) We must now decide how to treat the equality constraint. As we have no information about the problem that led to the model, we can have no good reason for choosing to replace it by two inequality constraints. We shall therefore substitute for x_1 or x_4. We choose x_4, for then we have only to substitute into one non-trivial and one trivial constraint, compared with one non-trivial constraint, one trivial constraint and the objective if we were to choose x_1. We have, from the equality constraint,

$$x_4 = 2x_1 - 8.$$

Substituting for x_4 in the (new) third non-trivial constraint gives

$$4x_1 + x_2 - x_3 - x_5 \geqslant 16,$$

and the non-negativity constraint for x_4 becomes

$$2x_1 \geqslant 8, \quad \text{i.e. } x_1 \geqslant 4.$$

(e) We must change the signs in the last (new) non-trivial constraint to give

$$-4x_1 - x_2 + x_3 + x_5 \leqslant -16,$$

and in $x_1 \geqslant 4$ to give

$$-x_1 \leqslant -4.$$

(f) Having replaced x_2 by $x_2 - x_5$ and eliminated x_4, we now have the four variables x_1, x_2, x_3, x_5. It can often help to reduce confusion, in such circumstances, to renumber the variables to run from 1 to 4, for example by relabelling x_5 as x_4. (There is less likelihood of confusion with small problems such as this one than with the large problems encountered in practice.) So, performing this renumbering, we get

maximize $z = -x_1 - 2x_2 + 2x_4$

subject to

$$\begin{aligned}
x_1 - x_2 - x_3 + x_4 &\leqslant -7\\
-4x_1 - x_2 + x_3 + x_4 &\leqslant -16\\
-x_1 &\leqslant -4\\
x_1, x_2, x_3, x_4 &\geqslant 0
\end{aligned}$$

In matrix notation this is:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [-1, -2, 0, 2]^T$, $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -4 & -1 & 1 & 1 \\ -1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -7 \\ -16 \\ -4 \end{bmatrix}.$$

1.11 We follow Procedure 1.3. From Solution 1.10, steps (a) to (c) result in:

maximize $z = -x_1 - 2x_2 + 2x_5$

subject to

$$\begin{aligned}
x_1 - x_2 - x_3 + x_5 &\leqslant -7\\
2x_1 - x_4 &= 8\\
x_2 - x_3 + 2x_4 - x_5 &\geqslant 0\\
x_1, x_2, x_3, x_4, x_5 &\geqslant 0
\end{aligned}$$

(d) All the signs need to be changed in the first constraint and the $\leqslant$ sign must be changed to a $\geqslant$ sign, giving

$$-x_1 + x_2 + x_3 - x_5 \geqslant 7.$$

(e) A surplus variable, x_6, must be subtracted from the new first constraint, giving

$$-x_1 + x_2 + x_3 - x_5 - x_6 = 7, \quad x_6 \geqslant 0.$$

The second constraint can be left as it is. A surplus variable, x_8, must be subtracted from the third constraint, giving

$$x_2 - x_3 + 2x_4 - x_5 - x_8 = 0, \quad x_8 \geqslant 0.$$

(f) We renumber the variables by relabelling x_8 as x_7, and we extend $\mathbf{c}$ to include zero elements for the surplus variables x_6 and x_7.

Thus, in matrix notation, the canonical form is:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [-1, -2, 0, 0, 2, 0, 0]^T$,
$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]^T$, $\mathbf{b} = [7, 8, 0]^T$,

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 1 & 0 & -1 & -1 & 0 \\ 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 2 & -1 & 0 & -1 \end{bmatrix}.$$

1.12 The objective does not change:

maximize

$$z = -9.6x_1 - 16x_2 - 16x_3 - 22x_4 - 19x_5 + 24x_6.$$

Adding slack and surplus variables to the inequality constraints, we get:

$$\begin{aligned}
x_1 + x_2 + x_3 + x_7 &= 10 \\
x_4 + x_5 + x_8 &= 8 \\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 5.6x_6 - x_9 &= 0 \\
1.2x_1 + 3.4x_2 + 8.0x_3 + 10.8x_4 + 8.3x_5 - 7.4x_6 + x_{10} &= 0 \\
x_1 + x_2 + x_3 + x_4 + x_5 - x_6 &= 0 \\
x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} &\geqslant 0
\end{aligned}$$

In matrix form this becomes:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [-9.6, -16, -16, -22, -19, 24, 0, 0, 0, 0]^T$,
$\mathbf{x} = [x_1, x_2, \ldots, x_{10}]^T$, $\mathbf{b} = [10, 8, 0, 0, 0]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1.2 & 3.4 & 8.0 & 10.8 & 8.3 & -5.6 & 0 & 0 & -1 & 0 \\ 1.2 & 3.4 & 8.0 & 10.8 & 8.3 & -7.4 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

1.13 Adding slack variables x_4, x_5, x_6, x_7, the canonical form, in matrix notation, is

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [30, 240, 500, 0, 0, 0, 0]^T$,
$\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]^T$, $\mathbf{b} = [25, 4000, 1000, 200]^T$,

$$\mathbf{A} = \begin{bmatrix} 0.01 & 0.2 & 1 & 1 & 0 & 0 & 0 \\ 20 & 120 & 240 & 0 & 1 & 0 & 0 \\ 0 & 20 & 30 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

1.14 We follow Procedure 1.1 to obtain a model.

(a) The objective is to maximize revenue. In view of the quantities involved, it would be wise to measure revenue in thousands of pounds.

(b) The variables are the numbers of each sort of hamper. Again, in view of the quantities involved, it would be sensible to work in thousands of hampers.

(c) The constraints are the quantities of each commodity in store. The parameters are the quantities of each item used in each hamper, the quantities of each commodity in store and the prices of the hampers.

(d) We shall use the following definitions:

z the total revenue, in thousands of pounds, from the sale of hampers;
x_1 the number of thousands of Hamper A made up;
x_2 the number of thousands of Hamper B made up;
x_3 the number of thousands of Hamper C made up.

(e)

	hamper			limit (thousands in store)
	A	B	C	
revenue (£)	50	30	25	—
nuts (kg)	6	4	8	10
puddings	3	2	0	4
sherry (bottles)	2	0	3	2
turkeys	1	1	0	1.5

(f) maximize $z = 50x_1 + 30x_2 + 25x_3$.

(g)
$$\begin{aligned}
6x_1 + 4x_2 + 8x_3 &\leqslant 10 \quad \text{(nuts)} \\
3x_1 + 2x_2 &\leqslant 4 \quad \text{(puddings)} \\
2x_1 + 3x_3 &\leqslant 2 \quad \text{(sherry)} \\
x_1 + x_2 &\leqslant 1.5 \quad \text{(turkeys)}
\end{aligned}$$

(h) $x_1, x_2, x_3, \geqslant 0$.

The model is already in standard form. Writing it in matrix notation gives:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [50, 30, 25]^T$, $\mathbf{x} = [x_1, x_2, x_3]^T$,

$$\mathbf{A} = \begin{bmatrix} 6 & 4 & 8 \\ 3 & 2 & 0 \\ 2 & 0 & 3 \\ 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 10 \\ 4 \\ 2 \\ 1.5 \end{bmatrix}.$$

To obtain the canonical form, we simply need to add a slack variable to each constraint, giving:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [50, 30, 25, 0, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2 \ldots, x_7]^T$,

$$\mathbf{A} = \begin{bmatrix} 6 & 4 & 8 & 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 10 \\ 4 \\ 2 \\ 1.5 \end{bmatrix}.$$

1.15 We follow Procedure 1.1 to obtain a model.

(a) The objective is to maximize daily takings, and these can be measured in pounds.

(b) The restrictions are on the numbers of plants treated per day, so we define the variables as the numbers of azaleas, rhododendrons and camellias treated per day.

(c) The constraints are the capacities for cutting, potting off and potting on, and also the marketing limitation on azaleas.

The parameters are the limits on capacity, the limit on the number of azaleas that can be sold and the selling prices.

(d) We shall use the following definitions:

z the daily takings, in pounds;
x_1 the number of azaleas treated per day;
x_2 the number of rhododendrons treated per day;
x_3 the number of camellias treated per day.

(e) Writing A for azaleas, R for rhododendrons and C for camellias, we have:

	A	R	C	upper limit (per day)
revenue (£)	3	5	4	—
cutting	✓	✓	✓	600
potting off	—	✓	✓	400
potting on	✓	✓	—	300
marketing	✓	—	—	80

(f) maximize $z = 3x_1 + 5x_2 + 4x_3$.

(g)
$$\begin{aligned} x_1 + x_2 + x_3 &\leqslant 600 \quad \text{(cutting)} \\ x_2 + x_3 &\leqslant 400 \quad \text{(potting off)} \\ x_1 + x_2 &\leqslant 300 \quad \text{(potting on)} \\ x_1 &\leqslant 80 \quad \text{(marketing)} \end{aligned}$$

(h) $x_1, x_2, x_3, \geqslant 0$.

(Notice that adding the potting off and marketing constraints yields $x_1 + x_2 + x_3 \leqslant 480$, so the cutting constraint will automatically be met when the potting off and marketing constraints are met. The cutting constraint is said to be *redundant*. We shall discuss redundant constraints in the next section.)

The model is already in standard form. Writing it in matrix notation gives:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [3, 5, 4]^T$, $\mathbf{x} = [x_1, x_2, x_3]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 600 \\ 400 \\ 300 \\ 80 \end{bmatrix}.$$

To obtain canonical form, we simply need to add a slack variable to each constraint, giving:

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [3, 5, 4, 0, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, \ldots, x_7]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 600 \\ 400 \\ 300 \\ 80 \end{bmatrix}.$$

1.16 We follow Procedure 1.1 to obtain a model.

(a) The objective is to minimize cost, which can be measured in pounds.

(b) The variables are the quantities of Special H and Super-Triple-X to be purchased. Given the quantities involved, these might best be measured in hundreds of litres.

(We do not need to include the quantity of the water-based product as a variable since we are told that its cost is negligible.)

(c) Three constraints are the minimum levels of Protein-Plus and Sheen required and the maximum level of conditioner allowed. There is also the constraint that only 1000 litres are to be made.

The parameters are the proportions of Protein-Plus, Sheen and conditioner in Special H and Super Triple-X, the limits on the levels of Protein-Plus, Sheen and conditioner, the limit on the total quantity to be manufactured, and the costs of Special H and Super Triple-X.

(d) We shall use the following definitions:

z the total cost, in pounds;
x_1 the quantity of Special H purchased, in hundreds of litres;
x_2 the quantity of Super Triple-X purchased, in hundreds of litres.

(e)

	Special H	Super Triple-X	limit (hundreds of litres)
cost (£ per 100 litres)	15	22.5	—
Protein-Plus (%)	20	5	$\geqslant 1$
Sheen (%)	20	50	$\geqslant 3$
conditioner (%)	20	30	$\leqslant 2.5$
total capacity	✓	✓	$\leqslant 10$

(f) minimize $z = 15x_1 + 22.5x_2$.

(g) Converting the percentages to decimals, the non-trivial constraints are:

$$\begin{aligned} 0.2x_1 + 0.05x_2 &\geqslant 1 \\ 0.2x_1 + 0.5x_2 &\geqslant 3 \\ 0.2x_1 + 0.3x_2 &\leqslant 2.5 \\ x_1 + x_2 &\leqslant 10 \end{aligned}$$

(h) $x_1, x_2 \geqslant 0$.

(If you included the water-based product as a variable, x_3 say, measured in hundreds of litres, the only change to the model would be that the fourth non-trivial constraint would become $x_1 + x_2 + x_3 = 10$.)

The general form of the model can be converted to standard form by making it a maximization and changing the signs of the first two constraints to give, in matrix notation:

maximize $z' = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $z' = -z$, $\mathbf{c} = [-15, -22.5]^T$, $\mathbf{x} = [x_1, x_2]^T$,

$$\mathbf{A} = \begin{bmatrix} -0.2 & -0.05 \\ -0.2 & -0.5 \\ 0.2 & 0.3 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -1 \\ -3 \\ 2.5 \\ 10 \end{bmatrix}.$$

To obtain canonical form, we can work directly from the general form, again making it a maximization and adding slack variables. We obtain, in matrix notation:

maximize $z' = \mathbf{c}^T\mathbf{x}$

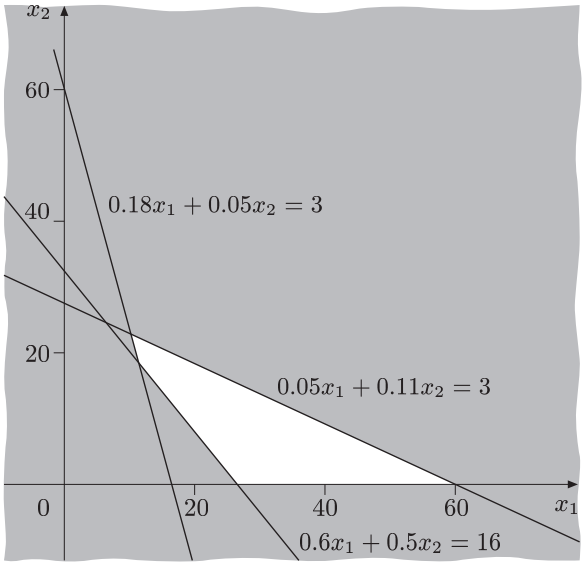
subject to $\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $z' = -z, \mathbf{c} = [-15, -22.5, 0, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, \ldots, x_6]^T$,

$$\mathbf{A} = \begin{bmatrix} 0.2 & 0.05 & -1 & 0 & 0 & 0 \\ 0.2 & 0.5 & 0 & -1 & 0 & 0 \\ 0.2 & 0.3 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 2.5 \\ 10 \end{bmatrix}.$$

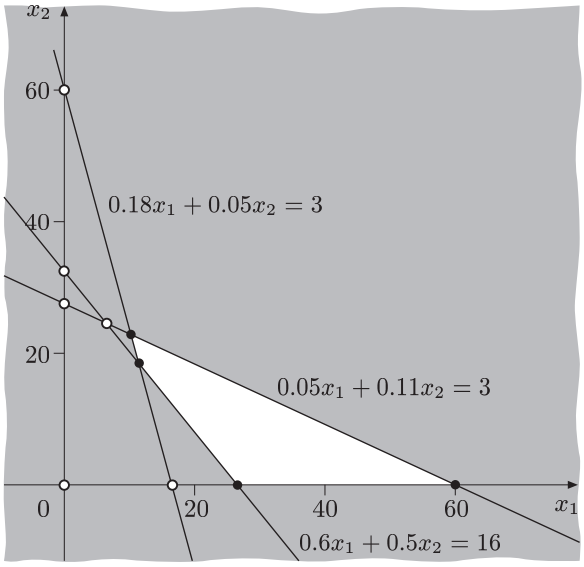
Section 2

2.1

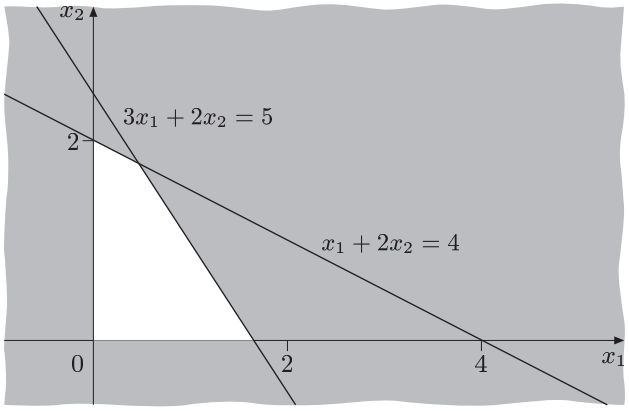


(Note that the constraint $x_1 \geqslant 0$ does not affect the feasible region — it is *redundant*.)

2.2 There are ten vertices, marked below. The vertices marked with a white dot lie outside the feasible region, while those marked with a black dot lie on its boundary and so are feasible vertices (four in all).

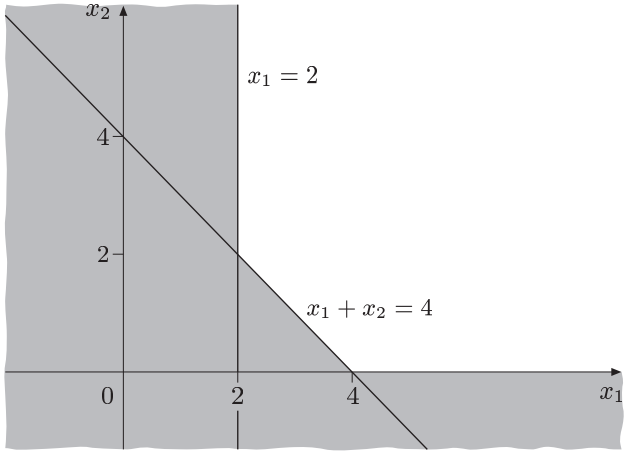


2.3 **(a)**



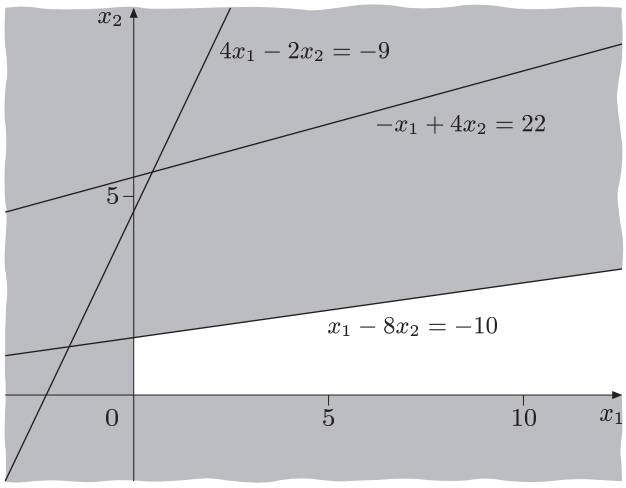
Bounded: a finite convex polygon.

(b)



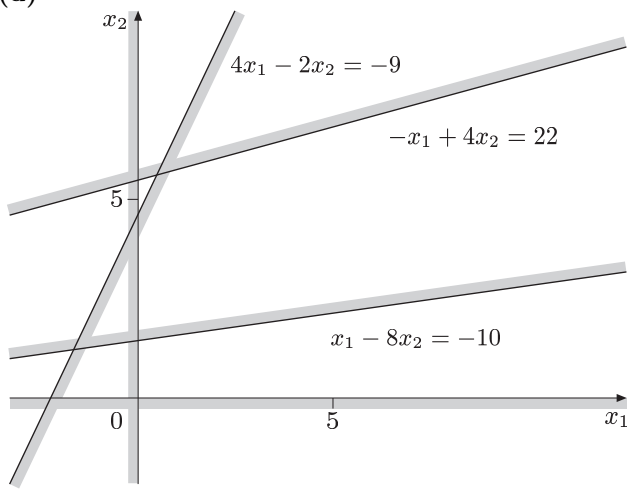
Unbounded: the values that both x_1 and x_2 can take are unbounded.

(c)



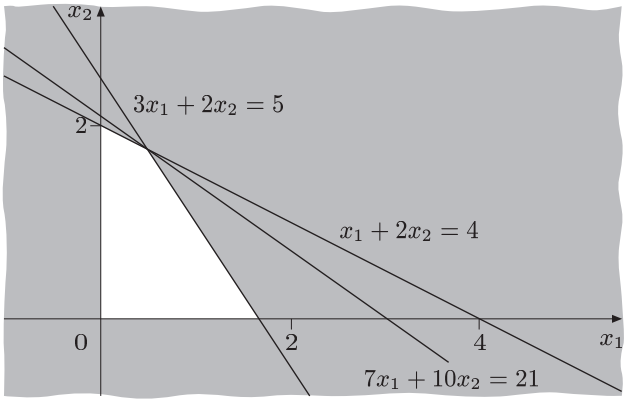
Unbounded: the values that x_1 and x_2 can take are unbounded.

(d)



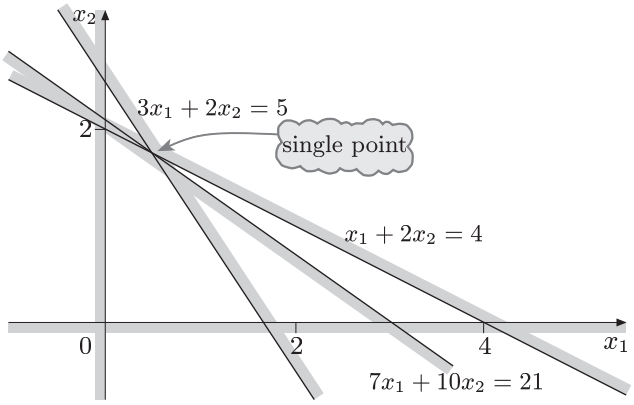
Empty.

(e)



Bounded: a finite convex polygon.

(f)



Bounded: a single point.

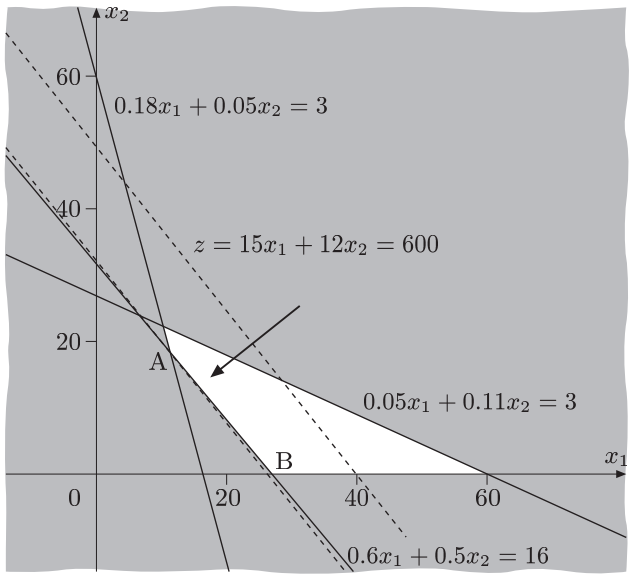
2.4 **(b)** As mentioned in the text, the constraint $x_1 \geqslant 0$ is redundant.

(c) Both the constraints $-x_1 + 4x_2 \leqslant 22$ and $4x_1 - 2x_2 \geqslant -9$ are redundant since the feasible region is unaffected if we remove either or both of them.

(e) The constraint $7x_1 + 10x_2 \leqslant 21$ is redundant. (Usually, where there is a degenerate vertex, at least one of the constraints is redundant.)

(f) The constraints x_1, $x_2 \geqslant 0$ are redundant.

2.5 **(a)**



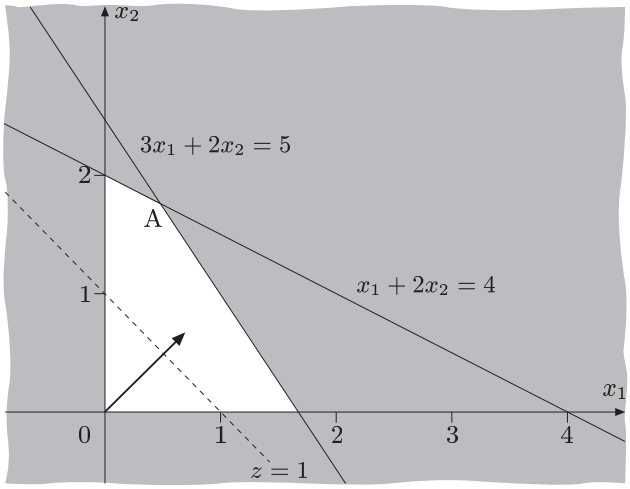
The figure shows the feasible region and the line $z = 600$. Here we are trying to *minimize* z, and the direction of *decreasing* z is marked by the arrow. (In this example, depending on the accuracy of your sketch, it may not be easy to judge whether the optimal vertex is at A or B. If this is the case, it is worth checking algebraically that the slope of the $z =$ constant lines is steeper than that of the line $0.6x_1 + 0.5x_2 = 16$.) The optimal vertex is at A, where the line $0.6x_1 + 0.5x_2 = 16$ crosses the line $0.18x_1 + 0.05x_2 = 3$. Thus the solution is at $(11\frac{2}{3}, 18)$. The value of the objective function at A is therefore $15 \times 11\frac{2}{3} + 12 \times 18 = 391$.

(b) The farmer should feed the ten pigs daily with $11\frac{2}{3}$ kg of fish meal and 18 kg of meat scraps at a daily cost of 391 p = £3.91.

(i) The protein and amino acids constraints are at their limits at the solution. So the pigs will be fed no more than their minimum requirements of each.

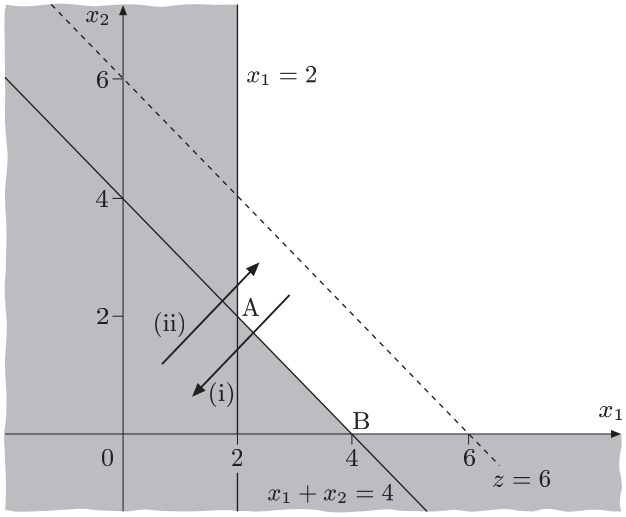
(ii) The calcium constraint is not at its limit. So the pigs will be fed less than the maximum permitted amount of calcium.

2.6 **(a)**



Unique optimizer at A, where $x_1 + 2x_2 = 4$ crosses $3x_1 + 2x_2 = 5$, i.e. at $x_1 = \frac{1}{2}$, $x_2 = 1\frac{3}{4}$ and $z = 2\frac{1}{4}$.

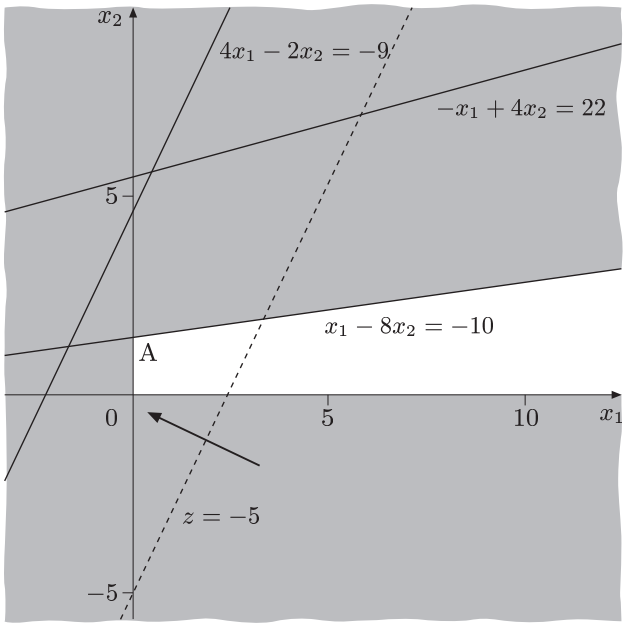
(b)



(i) *Multiple optimizers* along AB, where $x_1 + x_2 = 4$ and so $z = 4$.

(ii) *Unbounded solution*: z can be made as large as we please.

(c)

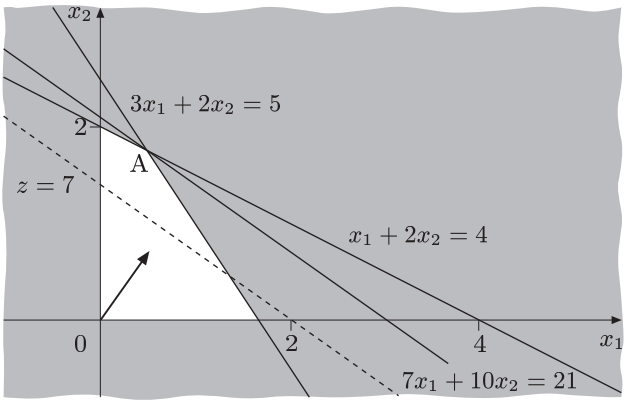


Unique optimizer at A, where $x_1 - 8x_2 = -10$ crosses $x_1 = 0$, i.e. at $x_1 = 0$, $x_2 = 1\frac{1}{4}$ and $z = 1\frac{1}{4}$.

(Note that, even though the $z =$ constant lines are parallel to the constraint bound $4x_1 - 2x_2 = -9$, we do not get multiple optimizers since the constraint $4x_1 - 2x_2 \geqslant -9$ is redundant.)

(d) *No solution*, as the feasible region is empty.

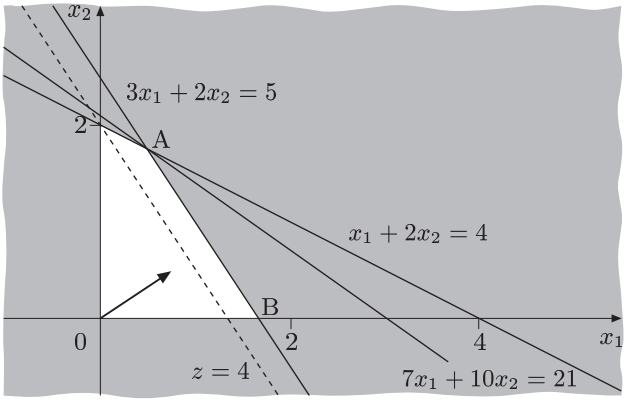
(e) (i)



Unique optimizer at A, where $x_1 + 2x_2 = 4$ crosses $3x_1 + 2x_2 = 5$. As in (a) this means $x_1 = \frac{1}{2}$ and $x_2 = 1\frac{3}{4}$, but now $z = 3\frac{1}{2}x_1 + 5x_2 = 1\frac{3}{4} + 8\frac{3}{4} = 10\frac{1}{2}$.

(Again note that, even though the lines $z =$ constant are parallel to the constraint bound $7x_1 + 10x_2 = 21$, we do not get multiple optimizers since the constraint $7x_1 + 10x_2 \leqslant 21$ is redundant.)

(ii)

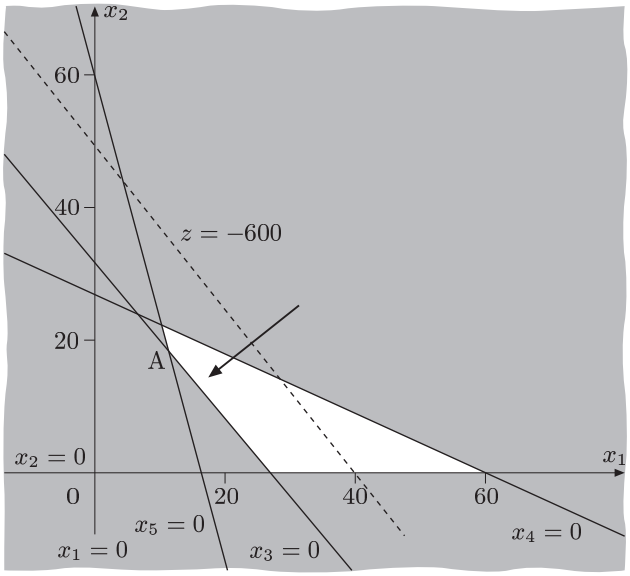


Multiple optimizers along AB, where $3x_1 + 2x_2 = 5$ and so $z = 5$.

(f) *Unique optimizer*: since there is only one feasible point there can only be one optimizer, at that point. The unique optimizer is thus at the point where $x_1 + 2x_2 = 4$ crosses $3x_1 + 2x_2 = 5$, i.e. at $x_1 = \frac{1}{2}$, $x_2 = 1\frac{3}{4}$ and $z = 5$.

(In (e)(i) and (f), you could equally well have obtained the coordinates of the optimizer by solving the pair of equations $x_1 + 2x_2 = 4$ and $7x_1 + 10x_2 = 21$ or by solving the pair $3x_1 + 2x_2 = 5$ and $7x_1 + 10x_2 = 21$.)

2.7



The constraint bounds are now interpreted as shown in the diagram. The lines $z =$ constant have negative values for the constants because we have changed the sign of z in changing the problem from a minimization to a maximization. The direction shown by the arrow is now that of *increasing* not decreasing z, and the optimal value of z (now a maximum rather than a minimum) will occur at the vertex A, as before. This is where $x_5 = 0$ crosses $x_3 = 0$. To find the values at the optimal vertex we solve

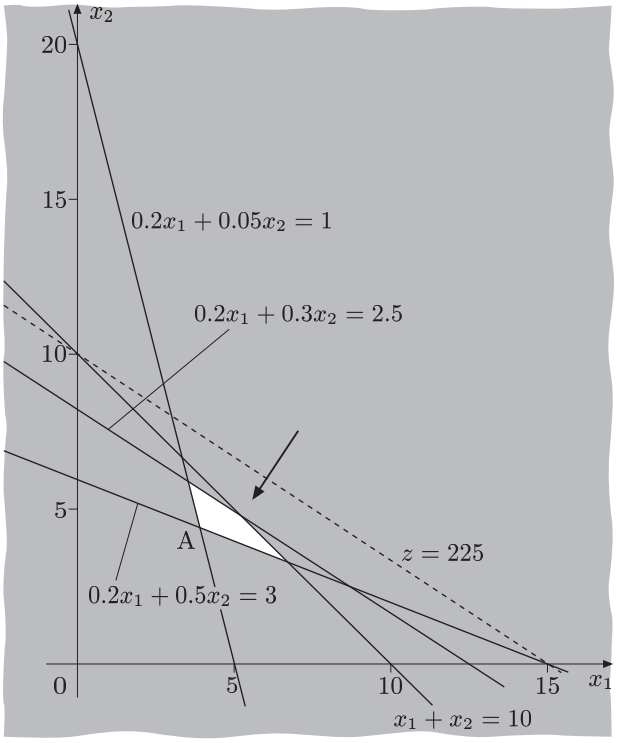
$$\begin{aligned} 0.6x_1 + 0.5x_2 &= 16 \\ 0.05x_1 + 0.11x_2 + x_4 &= 3 \\ 0.18x_1 + 0.05x_2 &= 3 \end{aligned}$$

giving $x_1 = 11\frac{2}{3}$, $x_2 = 18$, $x_4 \simeq 0.437$ and $z = 391$. These results can be interpreted as in Solution 2.5, but with the additional information that, at the solution, the pigs are being fed about 0.437 kg less calcium per day than the maximum permitted amount.

2.8 **(a)** The figure shows the feasible region and the line $z = 225$. Here we are trying to minimize z, and the direction of decreasing z is marked by the arrow. The optimal vertex is at A, where the line $0.2x_1 + 0.05x_2 = 1$ crosses the line $0.2x_1 + 0.5x_2 = 3$. Thus the solution is at $\left(\frac{35}{9}, \frac{40}{9}\right)$. The value of the objective function at A is $15 \times \frac{35}{9} + 22.5 \times \frac{40}{9} = 158\frac{1}{3}$.

(b) Since the optimal vertex is where the lines $0.2x_1 + 0.05x_2 = 1$ and $0.2x_1 + 0.5x_2 = 3$ cross, we can deduce that $x_3 = 0$ and $x_4 = 0$ at the solution. From the other two non-trivial constraints in the canonical form we obtain:

$x_4 = 2.5 - 0.2x_1 - 0.3x_2 = \frac{5}{2} - \frac{1}{5} \times \frac{35}{9} - \frac{3}{10} \times \frac{40}{9} = \frac{7}{18}$;
$x_5 = 10 - x_1 - x_2 = 10 - \frac{35}{9} - \frac{40}{9} = \frac{5}{3}$.



(c) The solution tells us that Hairways should purchase $\frac{35}{9} \times 100 \simeq 389$ litres of Special H and $\frac{40}{9} \times 100 \simeq 444$ litres of Super Triple-X in order to manufacture 1000 litres of shampoo, at a cost of £158.33. Also, the zero values for the surplus variables x_3 and x_4 tell us that the shampoo will contain the minimum 10% of Protein-Plus and the minimum 30% of Sheen. The value $\frac{7}{18} \simeq 0.39$ for the slack variable x_5 tells us that the shampoo will contain only about $(2.5 - 0.39) \times 10 = 21.1\%$ conditioner, well below the 25% limit. The value $\frac{5}{3} \simeq 1.67$ for the slack variable x_6 tells us that about 167 litres of the water-based product will be needed to make up the 1000 litres of shampoo.

Section 3

3.1 For point C we have:

- basic variables: x_2, x_3, x_5;
- non-basic variables: x_1, x_4;
- active constraints: $x_1 \geqslant 0$, $x_4 \geqslant 0$ (i.e. $\frac{2}{3}x_1 + 2x_2 \leqslant 16$);
- inactive constraints: $x_2 \geqslant 0$, $x_3 \geqslant 0$ (i.e. $2x_1 + 3x_2 \leqslant 30$), $x_5 \geqslant 0$ (i.e. $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$).

For point G we have:

- basic variables: x_1, x_2, x_3;
- non-basic variables: x_4, x_5;
- active constraints: $x_4 \geqslant 0$ (i.e. $\frac{2}{3}x_1 + 2x_2 \leqslant 16$), $x_5 \geqslant 0$ (i.e. $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$);
- inactive constraints: $x_1 \geqslant 0$, $x_2 \geqslant 0$, $x_3 \geqslant 0$ (i.e. $2x_1 + 3x_2 \leqslant 30$).

3.2 At D, the basic variables are x_1, x_3 and x_4 and the non-basic ones are x_2 and x_5; the active constraints are $x_2 \geqslant 0$ and $x_5 \geqslant 0$ (i.e. $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$). To move from D to A, we move along the edge DA. Along DA, the constraint $x_2 \geqslant 0$ is no longer active, while the constraint $x_5 \geqslant 0$ remains active, so x_1, x_2, x_3 and x_4 are non-zero and $x_5 = 0$. When we reach A, x_3 becomes zero, x_5 is still zero, and x_1, x_2 and x_4 are non-zero; the constraint $x_5 \geqslant 0$ remains active, and the constraint $x_3 \geqslant 0$ (i.e. $2x_1 + 3x_2 \leqslant 30$) becomes active. So, at A, the basic variables are x_1, x_2 and x_4 and the non-basic ones are x_3 and x_5; the active constraints are $x_3 \geqslant 0$ and $x_5 \geqslant 0$ (i.e. $2x_1 + 3x_2 \leqslant 30$ and $5\frac{1}{3}x_1 + 4x_2 \leqslant 64$).

3.3 **(a)** For the three-dimensional gardening model, whose canonical form has four non-trivial constraints and seven variables (the last four of which are slack variables), each vertex occurs at the intersection of three planes, on each of which $x_j = 0$ for some variable x_j ($j \in \{1, 2, \ldots, 7\}$). Thus, at each vertex, three variables are non-basic and the remainder are basic. The simplex method starts from the all-slack point, which is the basic feasible point where $x_1 = x_2 = x_3 = 0$. The method moves from one basic feasible point to an adjacent one along an edge, which is an intersection of two planes, and thus along which two of the variables are non-basic so that two of the constraints are active. As we leave one basic feasible point to move along an edge, we leave one of the planes (representing a constraint bound), and so one of the variables becomes basic, and that constraint becomes inactive. Then, when we reach the next basic feasible point, we meet a new plane (representing a new constraint bound), and so one other variable becomes non-basic, and the new constraint becomes active. The method continues moving from basic feasible point to adjacent basic feasible point until it reaches an optimizer.

(b) **(i)** $x_4 = 600$, $x_5 = 400$, $x_6 = 300$, $x_7 = 80$, $z = 0$;

(ii) $x_2 = 300$, $x_4 = 300$, $x_5 = 100$, $x_7 = 80$, $z = 1500$;

(iii) $x_2 = 300$, $x_3 = 100$, $x_4 = 200$, $x_7 = 80$, $z = 1900$;

(iv) $x_1 = 80$, $x_2 = 220$, $x_3 = 180$, $x_4 = 120$, $z = 2060$.

3.4 **(a)** With $\theta = 100$, from step (c) of Iteration 2 we have $x_2 = 300$, $x_3 = 100$, $x_4 = 200$ and $x_7 = 80$. Thus the new basic feasible point is $\mathbf{x} = [0, 300, 100, 200, 0, 0, 80]^T$, at which $z = 1900$.

(b) We need to express z solely in terms of the non-basic variables x_1, x_5 and x_6. This means eliminating x_3 from the expression for z derived in step (b) of Iteration 2, namely

$$z = 1500 - 2x_1 + 4x_3 - 5x_6,$$

and replacing it by an expression in terms of the non-basic variables. Now, from the potting-off constraint equation, and using the expression for x_2 derived from the potting-on constraint equation in step (b) of Iteration 2, we have

$$\begin{aligned} x_3 &= 400 - x_2 - x_5 \\ &= 400 - (300 - x_1 - x_6) - x_5 \\ &= 100 + x_1 - x_5 + x_6. \end{aligned}$$

Therefore

$$\begin{aligned} z &= 1500 - 2x_1 + 4(100 + x_1 - x_5 + x_6) - 5x_6 \\ &= 1900 + 2x_1 - 4x_5 - x_6. \end{aligned}$$

Thus $d_1 = -2$, $d_5 = 4$, $d_6 = 1$ and the only negative reduced cost is $d_1 = -2$, so x_1 becomes basic.

(c) Putting $x_1 = \theta$ and $x_5 = x_6 = 0$, we have

$$\begin{aligned} \theta + x_2 + x_3 + x_4 &= 600 \\ x_2 + x_3 &= 400 \\ \theta + x_2 &= 300 \\ \theta + x_7 &= 80 \end{aligned}$$

giving $x_2 = 300 - \theta$, $x_3 = 100 + \theta$, $x_4 = 200 - \theta$ and $x_7 = 80 - \theta$. As θ increases from zero, x_7 will therefore become zero first, when $\theta = 80$, so the constraint corresponding to $x_7 = 0$ (i.e. the marketing constraint $x_1 \leqslant 80$) becomes active first and x_7 becomes non-basic. The new basis list is $\boxed{4 \mid 3 \mid 2 \mid 1}$.

3.5 The canonical form is

$$\text{maximize } z = \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$$

where $\mathbf{c} = [2, -5, 3, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T$,

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 3 & -2 & 1 & 0 & 1 & 0 \\ 2 & -3 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 12 \\ 8 \\ 9 \end{bmatrix}.$$

Iteration 1, Step (a)

The all-slack point $\mathbf{x} = [0, 0, 0, 12, 8, 9]^T$ is feasible, and at this point $z = 0$. The basis list is $\boxed{4 \mid 5 \mid 6}$.

Iteration 1, Step (b)

$$z = 2x_1 - 5x_2 + 3x_3.$$

The reduced costs are $d_1 = -2$, $d_2 = 5$, $d_3 = -3$. The most negative is d_3, so x_3 becomes basic.

Iteration 1, Step (c)

If $x_3 = \theta$ while $x_1 = x_2 = 0$, we have

$$\begin{aligned} 3\theta + x_4 &= 12 \\ \theta + x_5 &= 8 \\ \theta + x_6 &= 9 \end{aligned}$$

giving $x_4 = 12 - 3\theta$, $x_5 = 8 - \theta$, $x_6 = 9 - \theta$. As θ increases from zero, x_4 becomes zero first, when $\theta = 4$. At this point, the constraint corresponding to $x_4 = 0$ becomes active and x_4 becomes non-basic. The new basis list is $\boxed{3 \mid 5 \mid 6}$.

Iteration 2, Step (a)

With $\theta = 4$, from step (c) of Iteration 1 we have $x_3 = 4$, $x_5 = 4$ and $x_6 = 5$. The new basic feasible point is $\mathbf{x} = [0, 0, 4, 0, 4, 5]^T$, at which $z = 12$.

Iteration 2, Step (b)

From the first constraint equation,

$$3x_3 = 12 - x_1 - 2x_2 - x_4,$$

so that

$$\begin{aligned} z &= 2x_1 - 5x_2 + (12 - x_1 - 2x_2 - x_4) \\ &= 12 + x_1 - 7x_2 - x_4. \end{aligned}$$

The reduced costs are $d_1 = -1$, $d_2 = 7$ and $d_4 = 1$. The only negative reduced cost is d_1, so x_1 becomes basic.

Iteration 2, Step (c)

If $x_1 = \theta$ while $x_2 = x_4 = 0$, we have

$$\begin{aligned} \theta + 3x_3 &= 12 \\ 3\theta + x_3 + x_5 &= 8 \\ 2\theta + x_3 + x_6 &= 9 \end{aligned}$$

giving $x_3 = \frac{1}{3}(12 - \theta)$, $x_5 = \frac{1}{3}(12 - 8\theta)$ and $x_6 = \frac{1}{3}(15 - 5\theta)$. As θ increases from zero, x_5 becomes zero first, when $\theta = \frac{3}{2}$. The constraint corresponding to $x_5 = 0$ becomes active and x_5 becomes non-basic. The new basis list is $\boxed{3}\boxed{1}\boxed{6}$.

The basic feasible point arrived at is $\mathbf{x} = \left[\frac{3}{2}, 0, \frac{7}{2}, 0, 0, \frac{5}{2}\right]^T$.

3.6 *Iteration 1, Step (a)*

The all-slack point, $\mathbf{x} = [0, 0, 0, 25, 4000, 1000, 200]^T$, is feasible, giving $z = 0$. The basis list is $\boxed{4}\boxed{5}\boxed{6}\boxed{7}$.

Iteration 1, Step (b)

$$z = 30x_1 + 240x_2 + 500x_3.$$

The reduced costs are $d_1 = -30$, $d_2 = -240$ and $d_3 = -500$. The most negative is d_3, so x_3 becomes basic.

Iteration 1, Step (c)

If $x_3 = \theta$ while $x_1 = x_2 = 0$, we have

$$\begin{aligned} \theta + x_4 &= 25 \\ 240\theta + x_5 &= 4000 \\ 30\theta + x_6 &= 1000 \\ x_7 &= 200 \end{aligned}$$

giving $x_4 = 25 - \theta$, $x_5 = 4000 - 240\theta$, $x_6 = 1000 - 30\theta$ and $x_7 = 200$. As θ increases from zero, x_5 becomes zero first, when $\theta = 16\frac{2}{3}$. The constraint corresponding to $x_5 = 0$ becomes active and x_5 becomes non-basic. The new basis list is $\boxed{4}\boxed{3}\boxed{6}\boxed{7}$.

Iteration 2, Step (a)

With $\theta = 16\frac{2}{3}$, from step (c) of Iteration 1 we have $x_3 = 16\frac{2}{3}$, $x_4 = 8\frac{1}{3}$, $x_6 = 500$ and $x_7 = 200$. The new basic feasible point is $\mathbf{x} = \left[0, 0, 16\frac{2}{3}, 8\frac{1}{3}, 0, 500, 200\right]^T$, at which $z = 8333\frac{1}{3}$.

Iteration 2, Step (b)

From the second constraint equation,

$$x_3 = \tfrac{1}{240}(4000 - 20x_1 - 120x_2 - x_5),$$

so

$$\begin{aligned} z &= 30x_1 + 240x_2 + \tfrac{25}{12}(4000 - 20x_1 - 120x_2 - x_5) \\ &= 8333\tfrac{1}{3} - 11\tfrac{2}{3}x_1 - 10x_2 - \tfrac{25}{12}x_5. \end{aligned}$$

The reduced costs are $d_1 = 11\frac{2}{3}$, $d_2 = 10$ and $d_5 = \frac{25}{12}$. Since these are all non-negative, the current basic feasible point is a solution.

A literal interpretation of this solution is that the smallholder should raise no hens, no pigs and $16\frac{2}{3}$ bullocks, for an income of £8333.33. Clearly, as he can't raise $\frac{2}{3}$ of a bullock, he should settle on 16 bullocks, say, for an income of £8000. (He can't raise 17, as these would cost him $17 \times$ £240 = £4080 to feed, and he only has £4000 available.) Raising $16\frac{2}{3}$ bullocks would take up all his money for food ($x_5 = 0$), but raising 16 leaves him with £160 to spare.

The fact that $x_4 = 8\frac{1}{3}$, $x_6 = 500$ and $x_7 = 200$ at the solution tells him that he has spare land and labour, and that none of the children's time for feeding hens is being used. In fact, with 16 bullocks he has 9 spare hectares and 520 spare hours, together with the children's time, so he might consider spending the spare £160 on feeding 8 hens, or on 1 pig and 2 hens. (In the first case he would still have 8.92 spare hectares and 520 spare hours, though the children would have some work to do; in the second case he would still have 8.78 spare hectares and 500 spare hours.)

(You will see how to solve problems in which some of the variables must take integer values in *Unit II.3*.)

Section 4

4.1 $\mathbf{x}_\mathrm{B} = [x_4, x_5, x_6, x_7]^T$;
$\mathbf{x}_\mathrm{N} = [x_1, x_2, x_3]^T$;
$\mathbf{c}_\mathrm{B} = [0, 0, 0, 0]^T$;
$\mathbf{c}_\mathrm{N} = [3, 5, 4]^T$;

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{N} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

$$_{\text{old}}\mathbf{x}_\mathrm{B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 600 \\ 400 \\ 300 \\ 80 \end{bmatrix} = \begin{bmatrix} 600 \\ 400 \\ 300 \\ 80 \end{bmatrix};$$

$$_{\text{old}}z = \mathbf{c}_{\mathrm{B}\,\text{old}}^T\mathbf{x}_\mathrm{B} = [0, 0, 0, 0] \begin{bmatrix} 600 \\ 400 \\ 300 \\ 80 \end{bmatrix} = 0.$$

4.2 $\mathbf{B}^{-1} = \mathbf{B} = \mathbf{I}$, and $\mathbf{c}_\mathrm{B} = \mathbf{0}$, so $\mathbf{c}_\mathrm{B}^T\mathbf{B}^{-1} = \mathbf{0}^T$.

$$\begin{aligned} z &= {}_{\text{old}}z - (\mathbf{c}_\mathrm{B}^T\mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_\mathrm{N}^T)\mathbf{x}_\mathrm{N} \\ &= 0 - \left([0, 0, 0, 0] \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} - [3, 5, 4]\right) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= 3x_1 + 5x_2 + 4x_3. \end{aligned}$$

So $d_1 = -3$, $d_2 = -5$, $d_3 = -4$ and thus x_2 should become basic.

4.3 In Solution 4.2 we saw that x_2 has become basic. The current basis list is $\boxed{4}\boxed{5}\boxed{6}\boxed{7}$,
$_{\text{old}}\mathbf{x}_\mathrm{B} = [600, 400, 300, 80]^T$ and $\mathbf{y}_2 = [1, 1, 1, 0]^T$.

The ratio test thus gives $\theta_1 = \frac{600}{1} = 600$, $\theta_2 = \frac{400}{1} = 400$, $\theta_3 = \frac{300}{1} = 300$ and $\theta_4 = \infty$, corresponding to x_4, x_5, x_6 and x_7 respectively. Thus the smallest non-negative ratio is $\theta_3 = 300$, which corresponds to $x_{\mathrm{B}3} = x_6$. So the variable x_6 becomes non-basic, and the constraint corresponding to $x_6 = 0$ becomes active.

4.4 For the all-slack point the basis list is $\boxed{4}\boxed{5}\boxed{6}$, and we have:

$\mathbf{x}_\mathrm{B} = [x_4, x_5, x_6]^T$, $\mathbf{x}_\mathrm{N} = [x_1, x_2, x_3]^T$;

$\mathbf{c}_\mathrm{B} = [0, 0, 0]^T$, $\mathbf{c}_\mathrm{N} = [2, -5, 3]^T$;

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & -2 & 1 \\ 2 & -3 & 1 \end{bmatrix}.$$

Iteration 1, Step (a)

$$_{\text{old}}\mathbf{x}_\mathrm{B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 12 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \\ 9 \end{bmatrix}.$$

$$_{\text{old}}z = \mathbf{c}_\mathrm{B}^T\, {}_{\text{old}}\mathbf{x}_\mathrm{B} = [0, 0, 0] \begin{bmatrix} 12 \\ 8 \\ 9 \end{bmatrix} = 0.$$

Iteration 1, Step (b)

$\mathbf{w}^T = \mathbf{c}_\mathrm{B}^T\mathbf{B}^{-1} = [0, 0, 0]$;

$$d_1 = \mathbf{w}^T\mathbf{a}_1 - c_1 = [0, 0, 0] \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} - 2 = -2,$$

$$d_2 = \mathbf{w}^T\mathbf{a}_2 - c_2 = [0, 0, 0] \begin{bmatrix} 2 \\ -2 \\ -3 \end{bmatrix} + 5 = 5,$$

$$d_3 = \mathbf{w}^T\mathbf{a}_3 - c_3 = [0, 0, 0] \begin{bmatrix} 3 \\ 1 \\ 1 \end{bmatrix} - 3 = -3.$$

The most negative reduced cost is d_3, so x_3 becomes basic.

Iteration 1, Step (c)

$\mathbf{y}_3 = \mathbf{B}^{-1}\mathbf{a}_3 = [3, 1, 1]^T$, $_{\text{old}}\mathbf{x}_\mathrm{B} = [12, 8, 9]^T$;

$\theta_1 = \frac{12}{3} = 4$, $\theta_2 = \frac{8}{1} = 8$, $\theta_3 = \frac{9}{1} = 9$.

The smallest non-negative ratio is $\theta_1 = 4$, which corresponds to $x_{\mathrm{B}1} = x_4$. So x_4 becomes non-basic and the constraint corresponding to $x_4 = 0$ becomes active. The new basis list is $\boxed{3}\boxed{5}\boxed{6}$.

For this new basis list we have:

$\mathbf{x}_\mathrm{B} = [x_3, x_5, x_6]^T$, $\mathbf{x}_\mathrm{N} = [x_1, x_2, x_4]^T$;

$\mathbf{c}_\mathrm{B} = [3, 0, 0]^T$, $\mathbf{c}_\mathrm{N} = [2, -5, 0]^T$;

$$\mathbf{B} = \begin{bmatrix} 3 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} 1 & 2 & 1 \\ 3 & -2 & 0 \\ 2 & -3 & 0 \end{bmatrix}.$$

Iteration 2, Step (a)

$$_{\text{old}}\mathbf{x}_\mathrm{B} = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & 0 & 1 \end{bmatrix} \begin{bmatrix} 12 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 5 \end{bmatrix}.$$

$$_{\text{old}}z = \mathbf{c}_\mathrm{B}^T\, {}_{\text{old}}\mathbf{x}_\mathrm{B} = [3, 0, 0] \begin{bmatrix} 4 \\ 4 \\ 5 \end{bmatrix} = 12.$$

Iteration 2, Step (b)

$\mathbf{w}^T = \mathbf{c}_\mathrm{B}^T\mathbf{B}^{-1} = [1, 0, 0]$;

$$d_1 = \mathbf{w}^T\mathbf{a}_1 - c_1 = [1, 0, 0] \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} - 2 = -1,$$

$$d_2 = \mathbf{w}^T\mathbf{a}_2 - c_2 = [1, 0, 0] \begin{bmatrix} 2 \\ -2 \\ -3 \end{bmatrix} + 5 = 7,$$

$$d_4 = \mathbf{w}^T\mathbf{a}_4 - c_4 = [1, 0, 0] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - 0 = 1.$$

The only negative reduced cost is d_1, so x_1 becomes basic.

Iteration 2, Step (c)

$$\mathbf{y}_1 = \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ -\frac{1}{3} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{8}{3} \\ \frac{5}{3} \end{bmatrix}, \quad _{\text{old}}\mathbf{x}_\mathrm{B} = \begin{bmatrix} 4 \\ 4 \\ 5 \end{bmatrix};$$

$\theta_1 = 4 \div \frac{1}{3} = 12$, $\theta_2 = 4 \div \frac{8}{3} = \frac{3}{2}$, $\theta_3 = 5 \div \frac{5}{3} = 3$.

The smallest non-negative ratio is $\theta_2 = \frac{3}{2}$, which corresponds to $x_{\mathrm{B}2} = x_5$. So x_5 becomes non-basic and the constraint corresponding to $x_5 = 0$ becomes active. The new basis list is $\boxed{3}\boxed{1}\boxed{6}$.

(You might like to compare the steps in this solution with those in Solution 3.5.)

4.5 The canonical form of the model obtained in Solution 1.14 is

maximize $z = \mathbf{c}^T\mathbf{x}$

subject to $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geqslant \mathbf{0}$

where $\mathbf{c} = [50, 30, 25, 0, 0, 0, 0]^T$, $\mathbf{x} = [x_1, x_2, \ldots, x_7]^T$,

$$\mathbf{A} = \begin{bmatrix} 6 & 4 & 8 & 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 10 \\ 4 \\ 2 \\ 1.5 \end{bmatrix}.$$

The all-slack point $\mathbf{x} = [0, 0, 0, 10, 4, 2, 1.5]^T$ is feasible. The rows of $\mathbf{A}$ are linearly independent. Hence, by Theorem 4.1, there is a basic feasible point that we can use to initiate the simplex method. (Indeed, the all-slack point is such a basic feasible point.) Furthermore, since we are told that the model does not suffer from degeneracy, we know, by Theorem 4.2, that the simplex method will terminate in a finite number of iterations. Finally, the restrictions on the quantities of each commodity in store means that the maximum number of hampers of each type is restricted. In other words, it is not possible to increase any of the variables indefinitely and so the model is bounded. The simplex method must therefore find a solution in a finite number of iterations.

Index