A. K. G. Heesterman

Matrices and Simplex Algorithms

A Textbook in Mathematical Programming and Its Associated Mathematical Topics

A. R. G. Heesterman
Department of Economics, University of Birmingham, U.K.

Matrices and Simplex Algorithms

A Textbook in Mathematical Programming and Its Associated Mathematical Topics

D. Reidel Publishing Company
Dordrecht : Holland / Boston : U.S.A. / London : England

Library of Congress Cataloging in Publication Data

Heesterman, A. R. G.
Matrices and simplex algorithms.

Includes index.
1. Programming (Mathematics) 2. Matrices. I. Title.
QA402.5.H43 1982 519.7 82-18540
ISBN 90-277-1514-9

Published by D. Reidel Publishing Company
P.O. Box 17, 3300 AA Dordrecht, Holland

Sold and distributed in the U.S.A. and Canada
by Kluwer Boston Inc.,
190 Old Derby Street, Hingham, MA 02043, U.S.A.

In all other countries, sold and distributed
by Kluwer Academic Publishers Group,
P.O. Box 322, 3300 AH Dordrecht, Holland

D. Reidel Publishing Company is a member of the Kluwer Group

Printed in The Netherlands

Introduction

This is a textbook devoted to mathematical programming algorithms and the mathematics needed to understand such algorithms. It was mainly written for economists, but the mathematics itself obviously has relevance for other disciplines.

It is a textbook as well as, in parts, a contribution to new knowledge. There is, accordingly, a broad ordering of climbing sophistication, the earlier chapters being purely for the student, the later chapters being more specialist and containing some element of novelty on certain points. The book is edited in five parts.

Part I deals with elementary matrix operations, matrix inversion, determinants, etc.

Part II is mainly devoted to linear programming.

As far as students' readability is concerned, these two parts are elementary undergraduate material.

However, I would claim, in particular with respect to linear programming, that I do things more efficiently than the standard textbook approach has it. This refers mainly to the search for a feasible solution i.e. Chapter 9, and to upper and lower limits, i.e. Chapter 10. I have also argued that the standard textbook treatment of degeneracy misses a relevant problem, namely that of accuracy.

In short, I would invite anyone who has the task of writing or designing an LP-code, to first acquaint himself with my ideas.

Parts III and IV are concerned with nonlinear programming. Part III gives the bulk of the theory in general terms including additional matrix algebra. It was obviously necessary to introduce definiteness at this point, but a full discussion of latent roots is refrained from. Proofs are therefore given as far as possible, without reference to eigenvalues. However, certain results will have to be taken on trust by those readers who have no prior knowledge on this point.

The main contribution to the literature made in part III, probably is Chapter 15, i.e. to explain both the first-order conditions and the second order conditions for a constrained maximum, in terms where one may expect the student to actually understand this admittedly difficult problem.

The unconventional concept of subspace convexity is not, and cannot be a true novelty; it is equivalent to the more usual way of formulating the second order condition for a constrained maximum in terms of determinants.

Part IV is concerned with quadratic programming. It does not give a comprehensive survey of algorithms. It gives those algorithms which I considered the most efficient, and the easiest to explain and to be understood.

With respect to novelty, Chapters 16 and 17 do not contain any original ideas or novel approaches, but some of the ideas developed in Chapters 9 and 10 for the LP case are carried over into quadratic programming. Chapter 19 does however, offer an algorithm developed by myself, concerning quadratic programming with quadratic side-conditions.

Part V deals with integer programming.

As in the QP case, the basic ingredients are taken from the existing literature, but the branching algorithm of section 20.2, although based on a well-established approach, was developed by myself. Also, the use of upper and lower limits on the lines of Chapter 10 proved particularly useful in the integer programming context.

Nothing in this book is out of the reach of undergraduate students, but if it is to be read in its entirety by people without prior knowledge beyond "O" level mathematics, the consecutive ordering of the material becomes essential and a two-year period of assimilation with a break between Part II and Part III would be preferable. However Parts IV and V will generally be considered to be too specialist on grounds of relevance and curriculum load, and accordingly be considered more suitable for postgraduate students specializing in O.R. or mathematical programming.

I have, however, myself used some sections of Chapter 16, not so much because undergraduates need to know quadratic programming for its own sake, but as reinforcement of teaching optimality conditions.

Concerning presentation, it may be observed that more than half of the number of pages is taken up by numerical examples, graphical illustrations and text-listings of programme-code.

The numerical examples and graphical illustrations are obviously there for purely educational purposes as are the student exercises.

The code-listings serve, however a dual purpose to back up the descriptions of algorithms, and to be available for computational use. Some tension between these two purposes is obviously unavoidable. I have made an effort to make the programme-texts readable, not only for the machine but also for the human reader, but if illustration for the benefit of the human reader were the only consideration I would have to cut down on the number of pages of code-listing much more than I have in fact done.

It appears to be appropriate to comment here on the use of the computer-language i.e. Algol 60, rather than the more widely used Fortran. While it is true that I simply know Algol very much better than Fortran, the choice appears also to be justified on the following intrinsic grounds:

The use of alphanumerical labels which are meaningful to the human reader, e.g.

PHASE I:, MAKE THE STEP:, etc., and
corresponding goto statements e.g.

'GOTO' PHASE I; helps to bridge the gap between programme description and programme-text in a way which is difficult to achieve by comment (or its Fortran equivalent) only.

Many procedure and programme texts also contain alphanumerical labels which are there purely for the human reader, as there are no corresponding goto statements.

While such labels are also possible in an "Algol-like" language as, for example Pascal, they cannot be used in Fortran. Furthermore, Fortran is simply more primitive than Algol.

I have made extensive use of block-structure and dynamic arrays and as a result my programmes don't require more core-space than is strictly necessary. Also, I gather that not all versions of Fortran permit recursive calls in the way I have used them in section 5.6 for the calculation of determinants and in section 20.4 for branching in integer programming.

The value of the text-listings as a direct source of ready-made programme-text is further compromised by the presence of warning-messages, not only in the main programmes but also inside the procedures.

The presence of these warning messages enhances the readability for the human reader but, as they are system-specific, they will in general require adaptation if the algorithms are to be applied in a different machine-environment.

Acknowledgement

I gladly acknowledge the use of the facilities of the University of Birmingham Computer Center, as well as the help of Dr. P. Soldutos, my student at the time, in setting up the private graph-plotting package used to make the graphs in this book.

University of Birmingham
21st May, 1982

A.R.G. Heesterman

Table of Contents

N.B. Answers to exercises are not included in this list of contents, but may be found in the places indicated with the exercises, usually at the end of the chapters.

Part I

MATRICES, BLOCK-EQUATIONS, AND DETERMINANTS

CHAPTER I

EQUATIONS-SYSTEMS AND TABLEAUX

1.1 Equations-systems

The following is an ordinary system of simultaneous linear equations:

$$0.867\,x_1 - 0.066\,x_2 = 240 \qquad (11)$$

$$-0.150\,x_1 + 0.850\,x_2 = 210 \qquad (12)$$

$$-0.167\,x_1 - 0.100\,x_2 + x_3 = 0 \qquad (13)$$

This system can be solved by performing certain operations:
Divide (11) by 0.867 (multiply by 1/0.867), and obtain:

$$x_1 - 0.076\,x_2 = 277 \qquad (21)$$

Multiply (21) by 0.150 and 0.167, respectively, and obtain:

$$0.150\,x_1 - 0.011\,x_2 = 42 \qquad (22)$$

$$0.167\,x_1 - 0.013\,x_2 = 46 \qquad (23)$$

Re-name (21) as (31), add (22) to (12) to become (32), add (23) to (13) to become (33), and obtain a new system:

$$x_1 - 0.076\,x_2 = 277 \qquad (31)$$

$$0.839\,x_2 = 252 \qquad (32)$$

$$-0.113\,x_2 + x_3 = 46 \qquad (33)$$

Divide (32) by 0.839 (multiply by 1/0.839) to obtain:

$$x_2 = 300 \qquad (42)$$

Multiply (42) by 0.076 and 0.113, respectively, to obtain:

$$0.076\,x_2 = 23 \qquad (41)$$

$$0.113\,x_2 = 34 \qquad (43)$$

Add (41) to (31), to become (51); re-name (42) as (52); add (43) to (33), to become (53), and obtain a new system:

$$x_1 = 300 \quad (51)$$

$$x_2 = 300 \quad (52)$$

$$x_3 = 80 \quad (53)$$

1.2 The use of a tableau

In section 1.1 three equations were written 5 times. Each time the variable-names x_1, x_2, and x_3 were written again. In total we did this 15 times.

We can economize on our writing effort, by writing the names of the variables only once. And if the procedure for obtaining a system of equations from its predecessor is a standardized one, we can dispense with explaining it every time. We could have done the job by writing 5 tableaux, as listed below:

x_1	x_2	x_3	=
0.867	-0.066	-----	240
-0.150	0.850	-----	210
-0.167	-0.100	1,000	---
1.000	-0.076	-----	277
0.150	-0.011	-----	42
0.167	-0.013	-----	46
1.000	-0.076	-----	277
-----	0.839	-----	252
-----	-0.113	1,000	46
-----	1.000	-----	300
-----	0.076	-----	23
-----	0.113	-----	34
1,000	-----	-----	300
-----	1.000	-----	300
-----	-----	1.000	80

CHAPTER II

MATRIX NOTATION

2.1 The purpose of matrix notation

Matrix notation provides a very compact way of describing certain well-defined numerical operations. As such it saves writing and reading effort in written communication about numerical operations. This applies to communication between one human being and another. It also applies to machine-programming. For most computers, there is by now a certain body of established programmes, routines, carrying out specific matrix- and vector operations. Reference to such routines saves programming effort. The use of matrix notation has also facilitated the analysis of numerical problems. This refers in particular to the properties of linear equation-systems. Such facilitation is really a corollary of the reduction in effort. Problems, which were formerly too complicated to grasp, now become manageable.

2.2 Some definitions and conventions

A matrix is a rectangular grouping of numbers, its elements. The elements of a matrix are grouped into a number of rows; each row containing the same number of elements, reading from left to right. Alternatively, we can say that the elements of a matrix are grouped into a number of columns, each column containing the same number of elements, reading from top to bottom.

Matrices often occur as tableaux, containing statistical information. For example:

British consumer's expenditure, Central Statistical Office: "National Income and Expenditure" (1967),

	1964	1965	1966
Coal and coke	273	269	257
Electricity	381	417	435
Gas	167	194	223
Other	58	60	58

Another application of tableaux (or matrices), was met in section 1.2: the arrangement of computations.

The order parameters of a matrix are two non-negative integer numbers. They are always listed in the same order. The first number indicates the number of rows of the matrix; the second order-parameter indicates the number of columns in the matrix. Normally, order-parameters will be positive numbers. But a more general validity of certain statements in matrix algebra may be obtained, if we admit the value of zero as a borderline case.

The number of elements in each row is the number of columns in the matrix. And the number of elements in each column is the number of rows in the matrix. It follows that the number of elements in the matrix is the product of its order parameters. A matrix of which the two order-parameters are equal is called a square matrix.

To indicate a matrix, we can use a letter. A capital letter is always used for that purpose. In printed text, a capital letter indicating a matrix, is generally given in heavy print.

We might for instance have:

$$C = \begin{bmatrix} 273 & 269 & 257 \\ 381 & 417 & 435 \\ 167 & 194 & 223 \\ 58 & 60 & 58 \end{bmatrix}$$

A corresponding lower case letter, with 2 indices will indicate an individual element of a matrix.
For example, $c_{3,2} = 194$.
The indices are always given in the same order: first the index indicating the row, then the index indicating the column.

Not all numerical information is suitably presented in a rectangular array. Suppose for instance we were interested only in total expenditure on fuel and light:

	1964	1965	1966
Expenditure on fuel and light	879	940	973

This is a vector. A vector is a matrix of which one of the order parameter is known to be unity. We distinguish between rows (matrices with only one row) and columns (matrices with only one column). The (total) fuel and light expenditure, was presented as a row. We could have presented it as a column as well.

To indicate a vector, we can use a letter. For that purpose, one always uses a non-capital letter. Vectors are normally indicated with italic print or heavy print, or in typescript - including photographically reproduced typescript, as in this book, with

underlining, to avoid confusion with indices. One should of course not use capital letters, this would create confusion with matrices.

Vectors will normally be assumed to be columns. When we want to indicate a row, this will be done by adding a prime to the letter.

The order of a vector is determined by listing the value of only one order-parameter. Elements of a vector are indicated with a small (non-capital) letter, without heavy print or underlining, with one index.

For instance, we might write: let $\underline{t}$ be a column-vector of order 3. We can also indicate the fact that a vector is a column (row) by stating its order as m by 1 (1 by n).

We may then define a (column) vector of consumer's expenditure on fuel and light, of order 3 (by 1).

$$\underline{t} = \begin{bmatrix} 879 \\ 940 \\ 973 \end{bmatrix}$$

The corresponding row-vector will be of order 1 by 3:

$$\underline{t}' = \begin{bmatrix} 879 & 940 & 973 \end{bmatrix}$$

The statement $\underline{t}'$ is a row-vector of order 1 by 3 is legitimate, but gives more information than is strictly needed. The 1966 figure can be indicated either as $t_3 = 973$, or as $t'_3 = 973$. The prime is quite superfluous here; hence we normally write $t_3 = 973$. Elements of a vector should always be indicated with their index. The use of ordinary small letters without index or heavy print (underlining), is conventionally reserved for variables or coefficients of an integer nature, such as indices and order-parameters.

Occasionally, one may also meet a single coefficient, which is not an element of a matrix or vector. Such a scalar is then indicated with a Greek letter. If required, a scalar can be interpreted as a matrix of order 1 by 1, as a column of order 1, or as a row of order 1.

A matrix which satisfies the property $a_{ij} = a_{ji}$ (and therfore obviously m = n) is called a symmetric matrix, e.g.

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

2.3 The transpose of a matrix

In paragraph 2 of this chapter, we met the following tableau:

British consumer's expenditure, at 1958 constant prices, on fuel and light:

	1964	1965	1966
Coal and coke	273	269	257
Electricity	381	417	435
Gas	167	194	223
Other	58	60	58

Now consider the tableau:

British consumer's expenditure, at 1958 constant prices, on fuel and light:

	Coal & coke	Elect-ricity	Gas	Other
1964	273	381	167	58
1965	269	417	194	60
1966	257	435	223	58

It will be observed, that this tableau gives exactly the same numerical information, as the previous one. But the presentation is different.

The corresponding matrices are said to be each other's transpose

$$C = \begin{bmatrix} 273 & 269 & 257 \\ 381 & 417 & 435 \\ 167 & 194 & 223 \\ 58 & 60 & 58 \end{bmatrix} \quad \text{and} \quad C' = \begin{bmatrix} 273 & 381 & 167 & 58 \\ 269 & 417 & 194 & 60 \\ 257 & 435 & 223 & 58 \end{bmatrix}$$

The transpose of a matrix is another matrix, with the rows of the first matrix as columns, and the columns of the first matrix as rows. It is conventional to indicate a transposition by a prime.

It follows that if a matrix A is of order m by n, then A' is of order n by m. The transpose of the transpose will have the rows of the transpose (= the columns of the matrix itself) as columns, and the columns of the transpose (= the rows of the matrix itself) as rows. The transpose of the transpose is the matrix itself, i.e.

$$(A')' = A$$