THE UNIVERSITY *of* EDINBURGH

COMPUTER CONSTRUCTION OF EXPERIMENTAL PLANS

by

MICHAEL FRANCIS FRANKLIN

Thesis submitted for the degree of

Doctor of Philosophy in the

University of Edinburgh,

1981

ACKNOWLEDGEMENTS

The research reported is this thesis is part of a program of work in the Agricultural Research Council Unit of Statistics on constructing a computer package for producing experimental plans to be used in agricultural experiments. I am grateful to the Agricultural Research Council for both financial support and facilities to undertake the research presented.

In two parts of the thesis the work reported is the result of collaboration. The contents of §3.5 and associated appendices were developed with Dr R.A. Bailey. I initiated the work and produced the basic solution and Dr Bailey improved the mathematical presentation. The computer program DSIGNX was developed with Mr A.D. Mann. The program contents, structure and principal algorithms were developed by me but the coding and implementation were performed by Mr Mann. Parts of §§10.3 and 10.4 are based on the (unpublished) DSIGNX programmers' manual which we prepared jointly.

My thanks are due to Dr H.D. Patterson for his encouragement and advice and to Professor D.J. Finney, Director of the Unit of Statistics for his support. I would also like to thank the secretarial staff of the Unit of Statistics for typing this thesis.

ABSTRACT

Experimental plans identify the treatment allocated to each unit and they are necessary for the supervision of most comparative experiments. Few computer programs have been written for constructing experimental plans but many for analysing data arising from designed experiments. In this thesis the construction of experimental plans is reviewed so as to determine requirements for a computer program. One program, DSIGNX, is described.

Four main steps in the construction are identified: declaration, formation of the unrandomized plan (the design), randomization and output. The formation of the design is given most attention. The designs considered are those found to be important in agricultural experimentation and a basic objective is set that the 'proposed' program should construct most designs presented in standard texts (e.g. Cochran and Cox (1957)) together with important designs which have been developed recently. Topics discussed include block designs, factorial designs, orthogonal Latin squares and designs for experiments with non-independent observations. Some topics are discussed in extra detail; these include forming standard designs and selecting defining contrasts in symmetric factorial experiments, general procedures for orthogonal Latin squares and constructing serially balanced designs.

Emphasis is placed on design generators, especially the design key and generalized cyclic generators, because of their versatility. These generators are shown to provide solutions to most balanced and partially balanced incomplete block designs and to provide efficient block designs and row and column designs. They are seen to be of fundamental importance in constructing factorial designs. Other

versatile generators are described but no attempt is made to include all construction techniques.

Methods for deriving one design from another or for combining two or more designs are shown to extend the usefulness of the generators. Optimal design procedures and the evaluation of designs are briefly discussed.

Methods of randomization are described including automatic procedures based on defined block structures and some forms of restricted randomization for the levels of specified factors.

Many procedures presented in the thesis have been included in a computer program DSIGNX. The facilities provided by the program and the language are described and illustrated by practical examples. Finally, the structure of the program and its method of working are described and simplified versions of the principal algorithms presented.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Outline of thesis

Satisfactory supervision of a comparative experiment requires that there exists a list of the treatments allocated to each experimental unit. We call such a list the experimental plan. In this thesis we review the construction of experimental plans with a view to determining requirements for a computer program to aid experimenters and consultant statisticians. We describe one such program DSIGNX. The review is not restricted to constructions available in DSIGNX but covers others that might be built into an extended or alternative program; we sometimes refer to this as the 'proposed' program.

We set the basic objectives that the proposed program should prove useful for constructing a wide range of plans for comparative experiments and that it should be compact. The need for a compact program is two-fold. Firstly, it should work on small interactive computers and secondly its use should not be too complicated to be learnt by an experimenter. To achieve both objectives we concentrate mainly on plans used in agricultural research and require that the program be able to reproduce most of the plans given by Cochran and Cox (1957), Davies (1978) and many more recent papers.

The simplest form of computer program is one in which the required plans are stored, unrandomized (the design), in a direct access file. Such a program is convenient for a limited range of designs but is not very versatile and rapidly becomes inefficient. At the opposite extreme, a program containing all known construction methods

would be versatile but bulky and difficult to use. We look for a few compact and simple methods of construction which between them yield the majority of useful designs but allow that a few designs are more easily stored than constructed.

We concentrate on three main methods of providing designs:

a) generating designs

b) modifying or combining existing designs

c) retrieving designs from libraries.

Two design generators are found to be very versatile and are used extensively throughout this thesis; they are

i) the design key generator (§3.3)

ii) the generalized cyclic generator (§§2.4 and 3.3).

They are closely related and both are useful for constructing factorial or pseudo-factorial designs. Illustrative examples are given in §1.1.1. A few other generators of special designs are also described (e.g. the optimal balanced designs in §4.4.2). There are many techniques for modifying one design or combining two designs to yield another. (Perhaps the best known of which is the construction of a lattice design with $r+2$ replicates from r orthogonal Latin-squares.) We describe some of the most useful of these methods.

The chapters of this thesis may be grouped as follows:

1; 2,3,4; 5,6; 7; 8; 9,10 .

The groups are largely independent and, except that Chapter 1 should be read first, may be read in any order; even within groups the chapters are largely independent. Chapter 1 provides an introduction to the remaining chapters and contains some definitions and procedures used later. Chapters 2-7 are concerned with methods of construction. Chapter 8 discusses design evaluation procedures which could be included in computer programs of the type being considered. Chapters

9 and 10 present the program DSIGNX which is in regular use for constructing experimental plans; the former chapter describes facilities provided and the latter programming aspects.

Most chapters are accompanied by appendices which perform three main functions:

i) they present extensions to current theory

ii) they contain constructions or tables of designs/ generating arrays.

iii) they contain examples or extra detail for topics discussed in the main text.

Appendix A9.4 should be read briefly, in conjunction with Appendix A9.3, before Chapters 2-10, as this appendix contains examples which could ease the understanding of other chapters. Other appendices may be omitted on first reading.

Chapter 2 is devoted to the construction of block designs and extends the range discussed in Chapters 9 to 13 of Cochran and Cox (1957). We discuss requirements for a versatile generator and show its relationship to several other generators in common use. The use of design key and generalized cyclic generators is described and we demonstrate that they provide solutions to virtually all BIB and PBIB designs of practical size. The construction of resolvable block designs (i.e. those which contain complete replicates) by these generators is also described and one construction - the α-series (§2.6.2.2) is found to be particularly important. Generators for row and column designs are also considered.

Chapters 3 and 4 describe the construction of factorial designs; they extend the constructions to be found in Chapters 5 to 8A of Cochran and Cox (1957). In Chapter 3 the importance and versatility of the design key procedure is demonstrated for a wide range of

fractional and confounded p^{n-k} designs. Two somewhat complementary problems are discussed in detail, the selection of a good all-round design for general use and the selection of a specific design. In the former case we present compact tables of design generators and in the latter a procedure to produce designs meeting the user's requirements. More procedures for constructing factorial designs are given in Chapter 4 where the discussion is widened to include other 2^n and 3^n designs and asymmetric factorial designs. It is found that simple generators, sometimes combined with procedures to delete or collapse factor levels, provide many useful designs.

Mutually orthogonal Latin squares (MOLS) have a special importance in the construction of other experimental designs; Chapter 5 is therefore devoted to simple methods of generating MOLS. Design key generators are shown to be useful for all squares of side $n \neq 4m + 2$. Simple cyclic generators are also considered; they complement the design key generators and the two jointly provide constructions for sets of MOLS of the largest known size for all cases $n \leq 30$, $n \neq 12, 24$. In Chapter 6 the construction of designs for experiments with interference between units is discussed; attention is restricted to change-over, superimposed and serially-balanced designs. Although some designs may be generated simply most are easier to store than to generate. We construct many serially balanced designs to form the basis of a suitable catalogue.

In Chapter 7 various methods of manipulating plans are described; these include operations on one plan to yield another and operations on two designs to yield a third. Randomization may be used to derive the experimental plan from a standard plan and some of the operations involved are closely related to those used in deriving one plan from another. For convenience we treat randomization as a

special method of manipulating plans.

Overall, the contents of chapters 2 to 7 show that relatively few simple procedures are required to form a versatile program. All designs (except chain-block designs) provided by Cochran and Cox (1957) are covered as well as several important extensions.

1.1.1 Some introductory examples

Four examples are presented to illustrate basic methods of construction used in this thesis; factor levels are taken as $0,1,\ldots,n_i-1$ where n_i is the number of levels of the ith factor.

Example 1.1 A 5×5 Latin square using a design key generator (§3.3).

Let the symbol applied to the intersection of the rth row and the cth column $0 \leq r, c \leq n-1$ be $3r+c$ reduced modulo 5 then the array yields a Latin square:

r	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4
c	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
t	0	1	2	3	4	3	4	0	1	2	1	2	3	4	0	4	0	1	2	3	2	3	4	0	1

Example 1.2 Cyclic generation of a balanced incomplete block (BIB) design (§2.5.1).

A BIB design for six treatments and 10 blocks with three experimental units in each is generated from initial blocks (0 2 3) and (2 3 5); cycling is modulo 5 and treatment 5 is invariant:

0	1	2	3	4		2	3	4	0	1
2	3	4	0	1		3	4	0	1	2
3	4	0	1	2		5	5	5	5	5

Example 1.3 Generalized cyclic generation of a resolvable block design (§2.6).

A resolvable design for 20 treatments in fifteen blocks of four

units can be formed by cyclic generation in increments of 4 from initial blocks (0 1 2 3)(0 5 10 15) and (0 17 14 11):

0	4	8	12	16	0	4	8	12	16	0	4	8	12	16
1	5	9	13	17	5	9	13	17	1	17	1	5	9	13
2	6	10	14	18	10	14	18	2	6	14	18	2	6	10
3	7	11	15	19	15	19	3	7	11	11	15	19	3	7

The same design may also be constructed by considering the treatments as combinations of a five-level and a four-level factor and cycling through levels of the first factor of the initial blocks (00, 01, 02, 03), (00, 11, 22, 33), (00, 23, 32, 41). We regard both as generalized cyclic constructions.

Example 1.4 Compounding two designs (§2.5.2 and §7.2.4).

Let D1 be a BIB design for three treatments in four blocks of three units:

0	1	2	3
1	2	3	0
2	3	0	1

and D2 be a single replicate design for eight treatments in four blocks of two units:

0	1	2	3
4	5	6	7

.

Replacing treatment i of D1 by the ith block of D2 yields the partially balanced incomplete block (PBIB) design for eight treatments in four blocks of six units:

0	1	2	3
4	5	6	7
1	2	3	0
5	6	7	4
2	3	0	1
6	7	4	5

1.2 Basic definitions

The following example assists in establishing the basic terminology used in this thesis:

Example 1.5 The effect of rate of application of nitrogen and phosphate on the yield of potatoes is to be estimated. Nine treatments comprising all combinations of three rates of nitrogen and phosphate are to be applied to 36 plots arranged in four blocks of nine plots each.

Following Cox (1958) the units of the smallest division of the experimental material such that any two may receive different treatments are called experimental units or more simply units; in the above example the plots form the units. Both units and treatments may be structured. Thus each unit above is defined by block and plot position within block and each treatment by the rate of nitrogen and phosphate applied. Following Nelder (1965) we call the structure of the units and treatments the block structure and treatment structure respectively; the factors which determine the structure are called block factors and treatment factors. Each factor may have one or more levels; these levels may be numeric or descriptive but are associated with formal levels 0, 1, 2, ...

In the experiments considered in this thesis at most one treatment may be allocated to a unit and a unit not receiving a treatment is excluded. An experimental plan is any list which identifies for each unit the treatment allocated. In producing experimental plans the units are commonly placed in lexicographical order i.e. some ordering of the block factors is selected and the units are then ordered by the formal levels of the factors, the levels of the last factor rotating fastest. The lexicographical ordering of the treatments in Example 1.5 is

{00, 01, 02, 10, 11, 12, 20, 21, 22}

Some procedures involved in the construction of an experimental plan are illustrated by the following example.

Example 1.6 We follow the procedure recommended by Fisher and Yates (1963, p.24) for constructing an experimental plan based on an 8×8 Latin square (chosen from an orthogonal set). The following stages may be recognized:

Stage 1. Select one of the seven orthogonal 8×8 squares from Table XVI, Fisher and Yates (1963). We call this the basic design or more simply the design to distinguish it from the squares derived from it. (The term design is used in this way by Clatworthy (1973)).

Stage 2. The rows and columns of the square are randomized. The choice of randomization identifies rows and columns of the design with rows and columns of the experimental plan and each location in the plan with a location in the design. Locations in the design are called unit labels by Franklin and Patterson (1978); locations in the plan are units (or, more accurately, unit identifiers). This stage thus corresponds to allocating unit labels to units.

Stage 3. The symbols of the square are randomized. The symbols of the design are called treatment labels by Franklin and Patterson (1978) and this stage corresponds to identifying treatment labels with treatments.

In this example units and unit labels have the same structure; similarly the treatment and treatment labels have the same structure. This is not always so. An 8×8 Latin square in Table XVI of Fisher and Yates (1963) (the basic design) can be constructed as follows:

1a) Identify the eight treatment labels, with combinations of three two-level pseudo-factors A, B and C:

0 → 000 1 → 001 2 → 010 3 → 011

4 → 100 5 → 101 6 → 110 7 → 111

1b) Form a basic column by a suitable permutation of the treatment labels.

1c) Form column j of the design (j = 0,1,...,7) by adding treatment label j to each treatment label in the basic column, where addition involves adding (modulo 2) the pseudo-factor representations e.g. 101 + 110 → 011.

The treatment labels are specified in terms of pseudo-factors and hence the structures of treatments and treatment labels differ. (Similarly, the structures of units and unit labels may differ.) The structure of the labels may also be affected by the use of restrictions as in the following example.

Example 1.7 An experimental plan is required for nine treatments, one of which is standard or control treatment, in six blocks of five units; the control treatment is to occur in each block. The treatments may be allocated in two steps as follows:

i) restrict attention to the control treatment and one unit per block;

ii) restrict attention to the remaining treatments and four units per block.

For the second step the eight treatment labels may be represented by three two-level pseudo-factors; a single nine-level treatment factor is thus associated with three two-level pseudo-factors.

We call the structure of the treatment labels the effective treatment structure and the structure of the unit labels the effective block structure. The effective structures may contain factors, pseudo-factors or both.

The relationship between block/treatment structures, effective block/treatment structures, restriction and pseudo-factors is discussed further in §9.4. Although we have distinguished between the structure of the plan and the effective structure of the design, for convenience the distinction is now dropped whenever there is no confusion and we use only the simpler terms.

1.3 Block and treatment structure

The use of formulae to define block and treatment structures is quite common (e.g. Claringbold (1969), Fowlkes and Lee (1971) and Alvey et al (1977)). Wilkinson and Rogers (1973) give the most general formulae but most of their extensions relate to forms for analysis. The most useful formulae for an experimental plan program are those presented by Nelder (1965) who defines just two operations, nesting and crossing, which have their usual meaning and denotes them:

$$B_1/B_2$$
$$B_1*B_2 \qquad 1)$$

He defines designs with simple block structure as those which can be represented by formulae involving nesting and crossing and have all Πb_i combinations of levels represented equally. He shows that designs with simple block structure have a valid randomization analysis and derives a null analysis of variance which partitions the analysis of variance into various different error strata based on different contrasts among the units.

In the notation of Wilkinson and Rogers (1973) the strata corresponding to the formulae 1) are given by the expansions

$$B_1/B_2 = B_1 + B_1.B_2$$
$$B_1*B_2 = B_1 + B_2 + B_1.B_2 \qquad 2)$$

More complicated formulae can be expanded in a similar manner (§7.3.2).

We find the following matrix to be important. Let $B_1, B_2 \ldots B_m$ be m block factors occurring in that order within a block structure formula, and define a lower triangular matrix M so that

$$m_{ii} = 1$$

$$m_{ij} = 1 \quad \text{if factor } i \text{ is nested in factor } j .$$

$$= 0 \quad \text{otherwise.}$$

We call this matrix a nesting matrix. The relationship between the block structure formula, nesting matrix and randomization is described in §7.3.2.

Example 1.8 A Latin square design with split plots has simple block structure (A*B)/C. Neither A nor B are nested within any other factor but C is nested in both A and B . The nesting matrix M is

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

The randomization sequence is randomize A , randomize B , then randomize C within each combination of levels of A and B .

Block and treatment structures determine the allowable permutations for deriving the plan from the design; they also as in 2) above determine how the degrees of freedom are allocated. Three other operations are used regularly in later chapters:

$$\begin{aligned} \text{dummy} &: \quad B_1 @ B_2 \\ \text{merge} &: \quad B_1 : B_2 \\ \text{product:} &\quad B_1 ; B_2 \end{aligned} \qquad 3)$$

Equating factor B to $B_1:B_2$ is equivalent to replacing factor B with b_1+b_2 levels by two factors with b_1 and b_2 levels. (Restriction (§1.2) is equivalent to replacing B by $B_1:B_2$ and

temporarily ignoring the levels of one factor.) If B_1 and B_2 are combined to form B we call the operation adding or merging . Equating factor B to $B_1;B_2$ is equivalent to replacing factor B with b_1b_2 levels by two pseudo-factors with b_1 and b_2 levels. Although we usually call B_1 and B_2 pseudo-factors we occasionally call them subfactors and call B a superfactor or the product of B_1 and B_2 . $B_1@B_2$ denotes that B_2 is a dummy factor introduced for convenience - usually to convert the design to a single replicate.

Example 1.9 A design for 25 treatments can be constructed by adding one treatment to a design for 24 treatments, itself constructed by a generalized cyclic generator based on a six-level and a four-level factor. The treatment set may therefore be represented by I:(A;B) where I, A and B are pseudo-factors with one, six and four levels respectively.

Operators (/,*,; etc) in structure formulae do not apply to effective structures and therefore designs with different block or treatment structures may have the same effective structures and even the same method of construction, e.g. quasi-factorial designs and partially confounded factorial designs. In several sections we use one design to yield experimental plans which are not equivalent under randomization.

1.4 Advantages of constructing experimental plans on a computer

Few computer programs have been written for constructing experimental plans and most have strictly limited objectives. On the other hand, programs for the statistical analysis of data from designed experiments have proliferated. Perhaps the prime reason for this discrepancy is the 'tedious' and 'repetitive' nature of standard analyses of a large data set. The value of computers for

data analysis stems from developments in both hardware and software. On the hardware side, the development of interactive terminals and, to some extent, the introduction of micro-processors have provided experimenters with easy access to computers. Increasingly, therefore, computers are seen as the principal method of storing data for analysis. On the software side the existence of programs such as GENSTAT (Alvey et al (1977)) and GLIM (Baker and Nelder (1978)) which are (relatively) easy to use and provide a wide range of statistical tools has encouraged experimenters to analyse their data with the aid of a computer. Similar developments also increase the usefulness of the computer in constructing experimental plans.

The advantage of using a computer in constructing experimental plans is dependent on the amount of work saved over doing the same job manually. Principal stages in the construction of an experimental plan, once the form of the experiment has been decided, are

i) construction of the design

ii) randomization

iii) presentation of the plan.

A computer program needs to provide significant assistance at one or more of these stages. At a non-statistical level, for example, a program may be justified if on input of the experimental plan the program supplied useful recording forms, field plans, etc. Mostly, the advantages of using a computer grow with increasing numbers of experimental units. For large trials the advantages are important; for small trials they may be trivial unless specification is very simple.

For complicated factorial trials construction of the design may be difficult or merely tedious. In either case computer construction has an important advantage, namely, the risk of clerical errors is

greatly reduced or eliminated. Clerical errors may arise from an operation being performed incorrectly or from transcription errors. In either case detection of the error may not be straightforward. A major clerical error can prove to be disastrous and transcription errors can seriously reduce the accuracy of the estimates of some important treatment contrasts. Computer construction of large designs can also save much labour at the randomization and presentation stages. For multiple experiments based on the same basic design, the advantages of using a computer come mainly from the randomization and presentation stages.

Perhaps the most important reason for computer construction of experimental plans is that it encourages experimenters to take more interest in the design of experiments. At present experimenters commonly modify their needs to suit available designs: computer construction helps provide a design to meet their needs. Furthermore, a suitable computer program can encourage more ambitious use of designs. Thus e.g. an α-series resolvable design (§2.6.2.2) may be preferred to a randomized block design or restricted randomization (Grundy and Healy (1950)) preferred to standard randomization. Furthermore, the use of a computer to construct the design, store and analyse data provides an integral process which encourages experimenters to plan ahead and extract maximum information from experiments.

CHAPTER 2

BLOCK DESIGNS

2.1 Summary

Useful methods are described for constructing incomplete block designs with a single treatment factor. We restrict attention to designs with equal replication of treatments and equal block size. The emphasis is on generators and simple methods of construction and we demonstrate that a wide range of useful block designs can be generated or stored easily within a small computer program.

In §2.2 basic terms are defined and a review of relevant literature given. A general construction for designs with simple block and treatment structure is defined in §2.3 and relationships with other constructions are noted. In §2.4 forms of cyclic and generalized cyclic generators are described. In §2.5 requirements for a program to construct balanced incomplete block designs are considered and extended to partially balanced incomplete block designs, attention being restricted to designs with triangular and Latin square association schemes.

Resolvable block designs are very important in agricultural experimentation and in §2.6 methods for constructing resolvable block designs are discussed, emphasis being placed on lattice designs and two series, called α and ϕ, of generalized lattice designs. Dual designs are discussed briefly. Two-replicate resolvable designs form a special case and are discussed separately.

Designs with crossed blocking structure are discussed in several sections of this thesis. In §2.7 resolvable row and column designs are considered with particular attention paid to lattice square designs and possible generalizations.

2.2 Definitions and a review of the literature

2.2.1 Definitions

A block design is an arrangement for v treatments in b blocks where the ith treatment is replicated r_i times and the jth block contains k_j units. (We regard the treatments as being the levels of a single treatment factor; designs with treatments having factorial structure are described in Chapters 3 and 4.) A block design with equal replication of all treatments is equireplicate and one with equal sized blocks is proper. A proper equireplicate block design with $b = v$, $r = k$ is symmetric. If all treatments occur at most once in a block the design is binary. If the blocks can be arranged into groups such that in each group all treatments are replicated q times the design is q-resolvable; a 1-resolvable design is resolvable. Each group of blocks is said to form a superblock. (A superblock is used throughout this thesis to denote a block containing all treatments equally often.)

The block structures of a block design and resolvable block design take the form

BLOCK/PLOT

and SUPERBLOCK/BLOCK/PLOT

respectively where each unit is defined by the levels of the block factors. A design with crossed block structure

ROW*COLUMN

where each unit receives exactly one treatment is called a row and column design. We reserve the term resolvable row and column design for one with block structure

SUPERBLOCK/(ROW*COLUMN).

In a multiple replicate design each treatment may be uniquely

identified by introducing a dummy replication factor; in a design where each treatment has r replicates this factor has r levels. The allocation of block and treatment factors level names being arbitrary, two designs are said to be isomorphic if one can be derived from the other by valid randomization of the units and treatments.

Two useful matrices are defined for block designs:

i) a $v \times b$ incidence matrix N whose (i,j)th element is the number of times the ith treatment occurs in the jth block;

ii) a $v \times v$ concurrence matrix NN^T whose (i,j)th element is, for binary designs, the number of times treatments i and j occur in the same block. The matrix $C = \underline{r}^{\delta} - N\underline{k}^{-\delta}N^T$, where r^{δ} and k^{δ} are diagonal matrices $\text{diag}(r_1 \ldots r_v)$ and $\text{diag}(k_1 \ldots k_b)$, plays an important role in the analysis of block designs (§8.2). C is the matrix of treatment sums of squares and products adjusted for blocks; we refer to it as the matrix of reduced normal equations. The generalized inverse of C determines the variance-covariance matrix of the estimated treatment effects. The related matrix $A = \underline{r}^{-1/2} C \underline{r}^{-1/2}$ where $r^{-1/2} = \text{diag}(r_1^{-\frac{1}{2}}, \ldots, r_v^{-\frac{1}{2}})$ we call the standardized matrix. Many desirable properties of block designs are determined by matrix C (and hence by A). The rank of C is at most $v-1$; if C has this rank the design is connected; in a connected design all treatment contrasts are estimable. A binary design is pairwise-balanced if all pairs of treatments occur equally often in the same block and variance-balanced if every normalized treatment contrast is estimated with the same variance.

The two forms of balance imply all off-diagonal elements of C are equal and for proper designs they are equivalent giving a balanced incomplete block (BIB) design. A partially balanced

incomplete block (PBIB) design is one in which i) the diagonal elements of C are equal and ii) the elements of any row and column can be obtained by permutation of the elements of any other row or column. (This definition is due to Pearce (1963); a definition in terms of association schemes is to be found in many standard texts (e.g. Raghavarao, 1971).)

In a PBIB design the variance of the difference of two treatment effects is dependent on the pair chosen. The number of distinct variances is dependent on the distinct roots of matrix C ; if C has m distinct latent roots the design is denoted a PBIB (m) design. The efficiency of a treatment contrast in an incomplete block design is its variance compared to that of a complete block design:

$$\frac{\text{variance in complete block design}}{\text{variance in incomplete block design}} = e \frac{\sigma_t^2}{\sigma_k^2}$$

where σ_t^2 and σ_k^2 are the variances of the units when arranged in complete blocks and incomplete blocks, respectively. The coefficient e is called the efficiency factor of the contrast. The mean efficiency of a design is an average of the contrast efficiencies; there are several alternative measures which are described in more detail in Chapter 8. One important measure of potential mean efficiency in a connected equi-replicate design is the harmonic mean of the latent roots of the standardized matrix A. We call this the harmonic mean efficiency factor; a design which has a maximum or near maximum value for this factor is said to be efficient.

Matrices N^T and $N^T N$ are the incidence and concurrence matrices respectively of the dual design obtained by interchanging the treatments and blocks. A proper equireplicate design and its dual share many efficiency properties (§8.2). For designs with more complicated block structure the dual operation is conveniently regarded

as an exchange of block and treatment factors. (§2.6.3).

Many designs in this chapter involve cyclic generation. In a cyclic design the treatments in an initial block are specified as residues modulo v , and the treatments in the jth block (j=0,1...v-1) are obtained by addition of j modulo v to the treatments in the initial block. In a generalized cyclic design the treatment factor is temporarily replaced by pseudo-factors and the blocks are formed by cyclic generation of one or more pseudo-factors. We define generalized cyclic designs to include cyclic designs and use cyclic generation to mean either form of generation as determined by the context.

2.2.2 Review of literature on block designs

The range of BIB designs likely to be used regularly in experimentation is covered by the collection for 15 or fewer replications presented by Fisher and Yates (1963). This collection based largely on the work of Yates (1936a), Bose (1939) and Rao (1961a) makes much use of cyclic generators. Fisher and Yates also list designs proved non-existent and those for which the solution was unknown in 1962. Few of the latter have since been solved. D.A. Preece (1981, private communication) lists various alternative constructions to those given by Fisher and Yates and also presents a useful bibliography.

PBIB designs originated with the lattice designs of Yates (1936b) and were formally defined by Bose and Nair (1939); Bose and Shimamoto (1952) redefine them in terms of association schemes. Bose and Nair (1939) present some methods of constructing PBIB(2) designs which are extended by Bose, Shrikande and Bhattacharya (1953). The early work is summarised in a catalogue of designs by Bose, Clatworthy and

Shrikande (1954). Significant extensions to the collection of PBIB(2) designs are made by Clatworthy (1955, 1956), Freeman (1957), and Chang et al (1964, 1965). A summary of this work and extensive tables are given by Clatworthy (1973).

Although designs with a cyclic association scheme form a very small class in the catalogues of Bose et al (1954) and Clatworthy (1973) cyclic generators have proved to be very powerful. Cyclic generators are given for most designs with a triangular association scheme by Masuyama (1965) and for some group divisible designs by Freeman (1976a). As with BIB designs, PBIB designs have inspired much mathematical research into the associated combinatorial and constructional problems; this work is described in Vajda (1967) and Raghavarao (1971).

BIB designs are the most efficient among all designs for the same number of treatments and block size: PBIB designs do not necessarily share this property and some are very inefficient. PBIB(2) designs have proved useful primarily because of ease of analysis and because there are only two standard errors for treatment differences. Their importance has diminished because of the use of computers for analysis and the development of designs which not only have, effectively, very few standard errors but are also very efficient. In the main, these designs can be constructed by cyclic generators.

Cyclic designs are used for paired-comparison trials by Kempthorne (1953) and for general block size k by David and Wolock (1965) and John (1966). David (1963) applies group theory to classify designs according to their initial blocks. The early work is summarised, and extensive tables of useful designs given, by John, Wolock and David (1972). John (1981) provides compact tables of initial blocks for designs which compare well with these.

A useful extension to cyclic designs is provided by generalized cyclic designs. The most important class is that with two treatment pseudo-factors; besides yielding solutions for many BIB and PBIB designs this class usually contains designs of maximum or near maximum efficiency. Jarrett and Hall (1978) discuss their usefulness.

Resolvable block designs are very important in agricultural trials for both managerial and statistical reasons; Patterson, Williams and Hunter (1978) outline some of their advantages. Lattice designs and rectangular lattice designs (Harshbarger (1949)) have been used extensively; both types are efficient but have severe restrictions on the number of treatments. Extensions to rectangular lattices allowing for designs with $s(s-k)$ treatments in blocks of $s-k$ units are noted by Harshbarger (1949) and Kempthorne (1952) but appear not to have been widely used. These designs are called generalized lattice - ϕ series by Patterson and Silvey (1980) who point out their usefulness in supplementing a second series of generalized lattice designs, the α series, in varietal trial work. The α series designs, which were introduced by Patterson and Williams (1976), are essentially generalized cyclic designs. The best of these designs are very efficient and often coincide with lattice or rectangular lattice designs. No catalogue of α series designs has been published but the paper by Patterson, Williams and Hunter (1978) contains a useful short table of wide applicability. Williams, Patterson and John (1976, 1977) discuss the construction of efficient two-replicate resolvable designs from symmetric block designs, and present some optimal constructions.

Row and column designs have received somewhat restricted attention, much of which has been concerned with extending Graeco-Latin squares to rectangles (e.g. Preece, 1971) and BIB designs to Youden squares.

Resolvable row and column designs have, apart from lattice square designs (Yates, 1937b), received little attention in the literature.

2.3 Generators for single replicate designs

In this section some requirements for a versatile generator for single replicate block designs are considered and the relationship with some standard generators is shown. The generator may be used for multiple replicate or fractional replicate designs by the introduction of a dummy treatment or block factor respectively. Some of the discussion is based on Patterson (1976) and, for convenience, the terminology and notation is similar. The term 'pseudo-factors' is used to denote factors which might be either standard factors or pseudo-factors.

Let there be m treatment (pseudo) factors T_i and n plot factors (i.e. block (pseudo) factors) P_j with numbers of levels t_i and p_j respectively. The total number of units and treatments is then

$$N = \Pi p_j = \Pi t_i \ .$$

It is required to find a one-to-one function allocating treatments to units. Expressing treatments and plots as vectors then

$$\underline{t} = f(\underline{p})$$

where f is a suitable integer function. If all possible vectors $\underline{t}$ and $\underline{p}$ are arranged in lexicographical order as the columns of an $m \times N$ matrix U and $n \times N$ matrix Q respectively then any operation which causes a permutation of the columns of U followed by identification of the kth column of U with the kth column of Q corresponds to a function in the set $\{f\}$. The functions vary in importance and some sets are isomorphic subject to valid randomization. (Randomization can be regarded as a function operating on the

columns of U.) Many useful designs can be constructed by row operations only and because there are few rows relative to columns these operations tend to be easier to express than column operations.

Matrices Q and U contain all plot and treatment vectors exactly once. Patterson (1976) calls them complete and matrices with columns in lexicographical order, base arrays. The base array for three factors Z_1, Z_2 and Z_3 with 3, 3 and 2 levels is:

Z_1	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2
Z_2	0	0	1	1	2	2	0	0	1	1	2	2	0	0	1	1	2	2
Z_3	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

A complete array (for plots) is to be converted to another (for treatments) by row operations. Now each row of a complete array is identified with the number of levels z_k of associated factor Z_k. Two types of row operation can be distinguished - those which do not affect the associated factor (A) and those which do (B).

Useful operations include:

A i) permutation of levels of factor Z_k.

B i) interchange two rows $\underline{x}_k$ and $\underline{x}_{k'}$. (This operation is of type A if $z_k = z_{k'}$.)

B ii) replace rows x_k and $\underline{x}_{k'}$ by a single row with elements $z_k x_{k'} + x_k$ in the residue set modulo $z_k z_{k'}$.

B iii) reverse operation B ii).

The operation A i) has two useful special cases:

A ii) add (modulo z_k) a constant c to each level of factor Z_k.

A iii) multiply (modulo z_k) each level of factor Z_k by a constant b which is co-prime to z_k.

If the levels of factors Z_j form a group under addition with residues modulo z_j then columns of the complete

array correspond to elements of the direct product group $\mathbb{Z}_1 \times \mathbb{Z}_2 \times \ldots \times \mathbb{Z}_n$. A complete array can be regarded as the direct product of complete sub-arrays formed by disjoint subsets $\{\mathbb{Z}_j\}$. Thus, for example, if levels of a subset of factors $\{\mathbb{Z}_b\}$ are fixed, then in columns corresponding to the restriction, the levels of the remaining factors form a complete sub-array (cf. the first six columns in the above example). In particular, the levels of each factor may be regarded as forming a complete array with levels of all other factors fixed.

Divide factors $\{\mathbb{Z}\}$ into two sets $\{\mathbb{Z}_a\}$ and $\{\mathbb{Z}_b\}$ then for each combination of levels of factors $\{\mathbb{Z}_b\}$ the combinations of levels for the factors $\{\mathbb{Z}_a\}$ form a complete sub-array. Applying operation A i) to sub-arrays converts the full complete array to another. This may be regarded as operation A i) conditional on $\{\mathbb{Z}_b\}$. A useful special case of this operation is:

A iv) add row x_j to row x_i and reduce modulo z_i.

Special cases:

i) Operation A ii) can be regarded as the addition of a constant vector to each column of the complete array, and operations A iii) and A iv) as multiplication on the left by non-singular diagonal and triangular matrices respectively. When the associated row factors have the same number of levels p then operation B i) is also of type A. As a set these operations allow the complete array Q to be modified by operations of the form $\underline{a} + K^T Q$ where $\underline{a}$ is a vector on K^T a non-singular matrix. (When p is not prime restrictions are placed on the permissible matrices and when the associated row factors have varying number of levels then associativity of matrix multiplication usually fails). These matrix operations form the basis for the

design key construction which is covered in detail in Chapter 3 and is also used in §2.6.1 for constructing lattice designs.

Example 2.1 Let the rows of a basic complete array correspond to factors ROW and COLUMN respectively, each factor having five levels. Pre-multiplication of the array by matrix $K^T = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}$ yields treatment combinations

A 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4

B 0 1 2 3 4 3 4 0 1 2 1 2 3 4 0 4 0 1 2 3 2 3 4 0 1

If COLUMNs form blocks then this array yields a single replicate design where AB^3 is confounded with blocks.

ii) The matrix operations in special case i) are also useful for constructing quasi-factorial designs. A lattice design with P^2 treatments, p prime, has three block factors, Superblock (P_1), Block (P_2) and Plot (P_3), and two treatment pseudo-factors together with a dummy replication factor. To form a multi-replicate design, the dummy replication factor may be identified with factor Superblock and the operations on the pseudo-factors made conditional on the level of Superblock.

Example 2.2 Up to five replicates of a lattice design for 25 treatments may be constructed by multiplying the basic array on the left by the matrix

$$K^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & q(P_1) \\ 0 & 0 & 1 \end{pmatrix}$$

where $q(P_1) = 0,1,2,3,4$ is the level of factor Superblock, i.e. the level of treatment pseudo-factors T_1 and T_2 are given by:

$$q(T_1) = q(P_2) + q(P_1) \cdot q(P_3) \mod 5$$

$$q(T_2) = q(P_3) \mod 5$$

Lattice designs are considered in more detail in §2.6.1

iii) The generalized lattice (α-series) designs (Patterson and Williams, 1976) are formed in much the same way as lattice designs but with the following rule:

$$q(T_1) = q(P_2) + f(P_1,P_3) \quad \text{mod } t_1$$

$$q(T_2) = q(P_3) \quad \text{mod } t_2$$

where $f(P_1,P_3)$ is an integer determined by the levels of P_1 and P_3. These designs are discussed in §2.6.2.2.

iv) The integer -1 is always coprime to the number of levels of the associated row factor. Thus the α-series generator in iii) can be extended by, for example, multiplying $q(P_2)$ by $g(P_1,P_3)$ where g takes the value ± 1. Williams (1975) used this form to construct a series of generalized block designs which he called the β series. (The series is not discussed in this thesis).

2.4 Cyclic and Generalized Cyclic Designs

2.4.1 Cyclic Designs

A useful description of cyclic designs is given in the tables of John et al (1972). We summarize those aspects of their description which relate to this thesis. A proper binary cyclic design for t treatments in t blocks of k units is formed as follows:

1) Select k distinct treatments $0 \leq t_s < t$ $(s = 1...k)$ to form an initial block.

2) Form the jth block $(j = 2,3...t)$ by addition of 1 to each treatment in the (j-1)th block followed by reduction modulo t.

There are $\binom{t}{k}$ ways of selecting the initial block but there are relatively few non-isomorphic designs under valid randomization operations for units and treatments. Two initial blocks are equivalent if they can be formed from each other by a combination

of the following operations.

i) addition (modulo t) of a constant to each element.

ii) reordering the elements in the block.

iii) any operation on treatments which permutes the elements of the vector $\underline{a}$ where a_d denotes the number of paired differences of size d in the initial block ($d = 1,2\ldots,[t/2]$).

One operation applicable under iii) is to multiply (modulo t) each treatment by an integer coprime to t. Denote the operation of multiplying by i as $R(t,i)$ $(o < i < t, (i,t) = 1)$ then $R(t,i)$ form a group under the operation *

$$R(t,i) * R(t,j) = R(t,ij \bmod t) .$$

For t prime the group is isomorphic to the multiplicative group of residues modulo t and is thus cyclic with primitive root $R(t,g)$, say. All possible isomorphisms between initial blocks can be established by repeated application of $R(t,g)$. For t non-prime the group is of order $\phi(t)$, Euler's function, and is normally not cyclic; it generates most but not all isomorphisms.

The concurrence matrix of a cyclic design is a symmetric circulant matrix (Appendix A8.5) whose first row is determined by the vector $\underline{a}$ defined in iii) above. For a design with h initial blocks (i.e. with ht blocks in all) the first row of the concurrence matrix is determined by the vector $\underline{a}_h = \Sigma \underline{a}_r$ where $\underline{a}_r$ is the vector associated with the rth initial block. (The equivalence classes associated with $\underline{a}_h$ can be derived by a simple cyclic procedure applied to the equivalence classes of the individual initial blocks.) The choice of design is determined by which equivalence class of initial blocks gives the design with the most desireable properties. Designs of high efficiency and near balance can often be obtained by

restricting attention to those equivalence classes which lead to designs with minimum variability among the off-diagonal elements. As $\underline{a}_h^T\underline{1} = hk(k-1)$ = constant this is equivalent to choosing the equivalence class for which $\underline{a}_h^T\underline{a}_h$ is smallest.

John et al (1972) present four tables containing initial blocks and efficiency factors for a wide range of designs. They give efficiencies of the differences between treatment effects and also elements of the first row of the generalized inverse of C the matrix of reduced normal equations. Three tables cover the following range:

A	$6 \leq t \leq 30$	$k \leq r \leq 10$	$k = 2$	$b \leq 150$
B	$6 \leq t \leq 15$	$k \leq r \leq 10$	$k > 2$	$b \leq 50$
C	$16 \leq t \leq 30$	$k = r$	$k > 2$	

A fourth table, D, contains designs for $r > k$ and is based on the use of factional sets but the designs of this table are generally less efficient than the best generalized cyclic designs so this table is of limited interest. The tables of initial blocks although bulky can be stored relatively easily on a computer. Alternative tables of initial blocks prepared by John (1981) are very much more compact and provide designs which are nearly optimal; these tables are unlikely to cause any space problems on small computers.

2.4.2 Generalized cyclic designs

Generalized cyclic generators can be used to construct both block designs and factorial designs (§§3.3 and 4.3.2), the block designs can be derived from the factorial designs by redefining the treatment structure. We present below a general generator, similar to that presented for factorial designs by John (1973), which is useful for for both types of design. We note also, however, the more restrictive but simpler form which is usually adequate for block designs and

we concentrate on this form.

When the number of treatments t is not prime the single treatment factor with t levels may be temporarily replaced by m pseudo-factors T_i with t_i levels $(i = 1...m)$ such that $\Pi t_i = t$. Let each treatment be represented by the m-tuple $a = a_1 a_2 ... a_m$ where $o \leq a_i \leq t_i - 1$ and define the addition of treatment combinations by

$$a_1 a_2 ... a_m + b_1 b_2 ... b_m = c_1 c_2 ... c_m$$

where $c_i = a_i + b_i \mod t_i$. Let γ be the lowest common multiple of the t_i and $\Delta_i = \gamma / t_i$. Let $w = \text{HCF}\ (\gamma, a_1\Delta_1, a_2\Delta_2 ... a_m\Delta_m)$ then the treatment vector $\underline{a}$ is a generator or primitive root of a cyclic abelian group of order γ/w with operation vector addition $+$.

A generalized cyclic design is constructed as follows:

1) Select initial block(s) of k treatments, each treatment represented by an m-tuple of pseudo-factor levels.

2) Select $u \geq o$ (linearly independent) treatments. For each treatment form the cyclic group then form the direct sum U of the cyclic groups. (The direct sum of two treatment sets $\{a_i\},\{b_j\}$ with only the zero treatment in common is the set formed by all treatments $a_i + b_j$.)

3) Form successive blocks by adding each treatment in U to all treatments in the initial block; repeat for each initial block.

For block designs the treatments selected under 2) usually contain one element unity and the rest zero; the operation then becomes that of cycling through all combinations of levels of a subset of u factors. The most rewarding application of generalized cyclic designs has been that for two pseudo-factors generated by cycling the levels of one factor only. This is the construction underlying the method of differences given by Bose et al (1953), the α-series of resolvable block designs of Patterson and Williams (1976)

and the generalized cyclic series of Jarrett and Hall (1978).

Example 2.3 Hall et al (1970) give a solution to the (previously unsolved) BIB design for $v = b = 56$, $r = k = 11$, $\lambda = 2$ by effectively forming two pseudo factors with 8 and 7 levels then cycling on the second factor with initial blocks

(00 14 24 31 41 51 62 64 65 66 71) (00 10 11 14 21 26 56 50 60 70 72)

(00 10 12 23 24 25 32 35 43 46 61) (00 20 23 31 33 36 54 55 63 72 73)

(00 22 26 40 41 42 52 55 61 74 76) (00 15 16 34 35 40 44 51 53 55 62)

(00 01 02 04 13 22 34 46 54 66 70) (00 13 15 30 36 43 45 60 71 74 75)

Generalized cyclic designs provide a useful extension to cyclic designs but like cyclic designs require careful selection of initial blocks: they provide useful extra designs for $b \geq v$ and superior designs for $b < v$.

2.4.3 A simple generalized cyclic generator

The following generalized cyclic generator is simple and versatile when used with techniques for restriction and pseudo-factors:

1) Form an array $\underline{s}$ of length m with ith element s_i, $1 \leq s_i \leq t_i$, the number of levels of the ith pseudo-factor;

2) Form initial block(s) containing k treatments each represented as levels of the m pseudo-factors.

3) Cycle over all factors, incrementing by 1 and treating the ith factor as having s_i levels. Any level in the initial block such that $\ell_i < 0$ or $\ell_i \geq s_i$ is treated as invariant. $b' = \prod_i^m s_i$ blocks are formed from each initial block.

Example 2.4 Fisher and Yates (1963) present a generalized cyclic construction for the BIB design with $v = 37$, $r = 12$, $k = 4$, $b = 111$. Their construction can be represented as follows. Form two treatment pseudo-factors A and B with four and 12 levels respectively where

only one treatment (3, 11) corresponds to level 3 of A. Generate the design from initial blocks:

(00 01 12 25) (01 03 08 10) (011 07 15 21) mod (3,11)

(311 00 10 20) mod (1, 11) (311 011 111 211) mod (1, 1)

The first three initial blocks each generate 33 blocks, the other two initial blocks generate 11 and one blocks.

Note: When $s_i = 1$ the associated levels are invariant and we sometimes replace 1 by - (e.g. mod (-,11)).

2.5 Balanced and partially balanced incomplete block designs

BIB and PBIB(2) designs are useful to both the experimenter requiring an incomplete block design and the statistician performing theoretical studies or constructing other designs; they are discussed by Cochran and Cox (1957, chapters 11, 13).

The object of this section is to determine how construction of BIB and PBIB designs can be performed by a computer program of reasonable magnitude. The aim is to please the experimenter rather than the mathematician and we are not concerned with non-existence theorems or with searching for solutions to unsolved problems. Where two or more distinct solutions exist only one is given; obscure constructions giving rise to one or two designs with special properties are ignored. However, because of their use in experimentation, resolvable designs and extended Youden square designs (designs for two-way control of heterogeneity) are preferred to other designs and occasionally presented as well as a simpler solution.

The construction of BIB designs is treated systematically by Fisher and Yates (1963) and the construction of PBIB designs by Clatworthy (1973). We do not repeat their work but see how it can be used in the proposed computer program. Most emphasis is given

to generalized cyclic methods of construction but some use is made of procedures, such as block section, which are also useful in constructing other designs. The number of distinct methods required is very small but, especially because of the requirements for resolvable and extended Youden square designs, there are several designs which are more efficiently stored than generated; we regard storage and retrieval of these designs as a method of construction.

2.5.1 Balanced Incomplete Block Designs

A BIB $(v, r, k, b; \lambda)$ design has v treatments, r replications, k units per block, b blocks, and λ concurrences for each pair of treatments. Only designs with $k \leq v/2$ are considered; designs with $k > v/2$ are complements of designs in this class, and designs with $k > v$ may be formed by merging (§7.2.4) incomplete and complete block designs. (The trivial design $v = b$, $r = k = 1$, $\lambda = 0$ is allowed.) The parameters satisfy two main conditions:

$$vr = bk$$

$$r(k-1) = \lambda(v-1) \ .$$

If $b \neq cv$ for some integer c then a design cannot be arranged as an extended Youden square design. If $v \neq sk$ for some integer s then a design cannot be resolvable. The conditions limit the availability of BIB designs, particularly when $v - 1$ is prime: the constraints on resolvable BIB designs are severe. In the range of interest $(r, k \leq 15)$ only a few designs, are both resolvable and extended Youden squares; they are all smaller BIB designs replicated.

The tables of Fisher and Yates (1963) provide a good starting point for a suitable computer program and the remainder of this section should be read in conjunction with their tables. They provide constructions for all solutions known in 1962 and of those

they marked unsolved, solutions have been found for only four designs, numbers 37 and 38 (Hall et al (1970)) and numbers 57 and 58 (Aschbacher (1971)); one, number 81, has been proved non-existent. Most constructions given by Fisher and Yates (1963) are generalized cyclic and usually provide extended Youden squares where possible; for $r, k \leq 10$ they provide a resolvable design where one exists.

The designs are divided into three main categories

i) unreduced designs

ii) orthogonal squares and geometric designs,

iii) combinatorial solutions.

They do not present solutions for all unreduced designs and do not indicate how these are best produced. For a general construction of unreduced designs with $r \leq 15$ only $k = 1$ or 2 need to be considered. For $k = 1$ the design is a trivial listing of treatments. For $k = 2$ a simple algorithm based on a cyclic generator is presented in Appendix A2.1 giving the unreduced design in a two-way layout. (When $v = 2m+1$ each column is a complete replicate and the design is an extended Youden square and when $v = 2m$ each row is a complete replicate and the design is resolvable.)

The orthogonal squares designs are of two types, those based on Euclidean geometries and those based on Projective geometries. The former can be derived from the latter by suitable block section and the operation can be reversed. However, over the range $r, k \leq 15$ the Euclidean designs (which are balanced lattice designs) are best generated using the design key procedure outlined in §2.6.1, while the Projective geometry designs may be constructed as cyclic designs. Besides the orthogonal square constructions there are four Euclidean designs, referenced 6(=E(3,2):2), E(3,3):1, E(3,3):2 and E(4,2):3, which are also best constructed using the design key procedure.

The associated Projective geometry designs 7(=P(3,2):2), P(3,3):1, P(3,3):2, P(4,2):3 together with P(4,2):1 can all be constructed using cyclic or generalized cyclic generators.

The majority of combinatorial solutions are based on generalized cyclic constructions. Of the 12 designs not constructed in this way, seven (1,3,16,17,22,26,46) are obtained by block section and the remainder (7,20,25,27,49) are presented in full or the constructions given are somewhat messy. The solutions for designs 7 and 25 are resolvable: those for designs 20 and 27 (Fisher and Yates (1963, p28)) are extended Youden squares based on ad hoc constructions. Alternative generalized cyclic constructions are noted in Appendix A2.2, together with a solution for BIB(14,13,7,26;6) not recorded by Fisher and Yates.

Of those designs with $b = cr$ the only solutions given by Fisher and Yates which are not extended Youden squares are designs 48, 49 and 67; of those with $v = sk$ only designs 34, 36, E(3,3):1, P(3,3):1, 61, 87 are not resolvable. The alternative constructions for designs 7, 48, 49, 34, E(3,3):1, P(3,3):1 given in Appendix A2.2, are therefore preferred. We are unaware of resolvable solutions to designs 36, 61 and 87 - those for 36 and 87 are 2-resolvable- and recommend that the Fisher and Yates solutions be kept. Design 67 could probably be converted to an extended Youden square.

Summary Generalized cyclic, design key or block section procedures yield convenient solutions for all BIB designs $r, k \leq 15$. The solutions given in Appendix A2.2 for designs 7, 48, 49, 34, E(3,3):1, P(3,3):1 are preferred to those of Fisher and Yates. If resolvable or extended Youden square designs are required solutions for designs 20, 25, 27 and 67 should be stored complete.

2.5.2 Partially Balanced Incomplete Block Designs

A PBIB(2) $(v, r, k, b; \lambda_1, \lambda_2)$ design has v treatments, r replications per treatment, k units per block and b blocks, each pair of treatments occur together in a block λ_1 times (first associates) or λ_2 times (second associates). The parameters satisfy the conditions

$$rv = bk$$

$$r(k-1) = n_1\lambda_1 + n_2\lambda_2$$

$$n_1 + n_2 = v - 1$$

where n_1 and n_2 are the number of first and second associates respectively. (For a more complete definition and properties see Raghavarao (1971)).

Clatworthy (1973) presents tables for over 950 PBIB (2) designs classified by association scheme; the most important schemes are:

Group divisible designs (Regular, semi-regular, singular)

Latin square designs

Triangular designs.

In this section we look for simple methods for constructing one design for each set of parameters listed by Clatworthy and show that with few exceptions two methods suffice. The designs not covered were originally constructed by a variety of procedures including trial and error but they are readily stored in a direct-access file.

For brevity, we restrict attention to designs with Latin square and triangular association schemes. Designs with Latin square $L_i(n)$ association schemes have n^2 treatments and concurrences defined by a set of $i - 2 \geq 0$ orthogonal Latin squares. The n^2 treatments are arranged in a square array and $i - 2$ orthogonal Latin squares superimposed on the array. Two treatments are first associates if they appear in the same row, column or have the same symbol in any of the

squares. Each treatment has $n_1 = i(n-1)$ first associates and $n_2 = (n-i+1)(n-1)$ second associates. Designs with a triangular $T(n)$ association scheme have $n(n-1)/2$ treatments. The treatments are arranged in an $n \times n$ array such that the array is symmetric about the leading diagonal which is empty: each treatment occurs twice. Two treatments are first associates if they occur in the same row. Each treatment has $2(n-2)$ first associates and $(n-2)(n-3)/2$ second associates. The standard form for $L_2(5)$ and $T(5)$ association schemes are

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

.	1	2	3	4
1	.	5	6	7
2	5	.	8	9
3	6	8	.	10
4	7	9	10	.

For both schemes any two treatments in the same row or column are first associates.

Fundamental designs A design derived from an association scheme by forming blocks (of the largest size) of mutual first associates we call a fundamental design. The fundamental design for a group divisible scheme for $v = n_1 n_2$ treatments and $n_1 - 1$ first associates is an $n_1 \times n_2$ array with columns forming blocks; that for a triangular $T(n)$ scheme is a two-replicate design with n blocks of $n-1$ units, each block corresponding to one row of the association scheme; that for a Latin square $L_r(n)$ scheme is a lattice design (§2.6.1) with r replicates. Many of the standard constructions for PBIB(2) designs are based on manipulations of the fundamental design. Several operations when applied to a fundamental design (or a design with the same association scheme) yield another design with the same scheme. Apart from the complement operation

the most useful operations are:

A. Replace each block by a BIB design containing the treatments of that block.

B. Let the PBIB design be q-resolvable in sets of s blocks; in a BIB design for s treatments replace the ith treatment by the ith block of the PBIB design. Repeat for each super-block.

C. Let A be a PBIB(2) design for t treatments in b_1 blocks of k units, and B be a BIB design for t treatments in b_2 blocks of k units. Then merge the blocks to form a design with $b_1 + b_2$ blocks. Alternatively B may be a PBIB(2) design with the same association scheme as A.

D. i) Form a design E(i) in which the jth block contains all ith associates of the jth treatment for fixed choice of i

ii) Supplement D i) by including treatment j in the jth block.

Clatworthy (1967) defines four families (labelled A-D) for designs with Latin square association schemes. The above definitions are extensions of his. Constructions A and B are special cases of the compound operation (§7.2.4) and construction C a special case of the merge operation (§7.2.4).

Example 2.5 Select the fundamental design for a group-divisible association scheme with n_2 blocks of n_1 units. Applying a BIB $(n_2, r, k, b; \lambda)$ design to blocks under operation B yields a singular group divisible design with parameters $(v, r, n_1k; r, \lambda)$. All singular group divisible designs can be so constructed.

Latin square schemes: We note firstly two trivial restrictions in our approach compared to that of Clatworthy (1973). Firstly, if an $L_i(n)$ scheme is based on a complete set of Latin squares the same design can also be categorised as an $L_{n+1-i}(n)$ scheme merely by interchanging first and second associates. Secondly for $i \geq 3$,

different $L_i(n)$ schemes may be formed by varying the choice of Latin squares. In both cases we ignore the distinctions.

The designs presented by Clatworthy were constructed by a variety of methods. In Appendix A2.3 the most important methods are listed and one is identified for each set of design parameters; six methods suffice. Of the six methods four can be replaced by use of a generalized cyclic generator. The remaining methods are A and C noted above. Both these methods can also be replaced by generalized cyclic generators but they are easily built into a computer program (§7.2.4) and provide enough designs to warrant being included.

Triangular schemes: There is a close relationship between the triangular $T(n)$ and Latin square $L_2(n)$ schemes. For $n = 2m - 1$ construct orthogonal Latin squares (§5.5) with elements $a_{ij} = m(i + j)$ and $b_{ij} = (j - i) + (n - 1)$ $(0 \leq i,j \leq n - 1)$ and construct an array with elements $a_{ij} + n\, b_{ij}$ (or, if preferred, $n\, a_{ij} + b_{ij}$). The resulting array forms a $L_2(n)$ association scheme very suitable for cyclic generation. Deleting the diagonal and 'folding-over' the array yields a triangular association scheme suitable for cyclic generation.

Example 2.6 Latin squares

0	3	1	4	2	and	4	0	1	2	3
3	1	4	2	0		3	4	0	1	2
1	4	2	0	3		2	3	4	0	1
4	2	0	3	1		1	2	3	4	0
2	0	3	1	4		0	1	2	3	4

give Latin square $L_2(5)$ and Triangular $T(5)$ association schemes:

20	3	6	14	17		.	3	6	9	2
18	21	4	7	10		3	.	4	7	5
11	19	22	0	8		6	4	.	0	8
9	12	20	23	1		9	7	0	.	1
2	5	13	21	24		2	5	8	1	.

For $n = 2m$ the association scheme is modified by bordering the array to the right and below by the vector $(a, a+1, \ldots b)$ where $a = (n-1)(n-2)/2$ and $b = n(n-1)/2 - 1$ (c.f. §5.5).

The close relation between triangular $T(n)$ and Latin square $L_2(n)$ designs and the fact that virtually all of the latter designs are readily constructed by cyclic methods suggest that the triangular designs may also be constructed in this way. Indeed, Masuyama (1965) has given cyclic constructions for most triangular designs based on association schemes closely related to those given above.

In Appendix A2.4 constructions for the PBIB designs with triangular association scheme are given. As with the Latin square designs only about six methods are required and these can be readily reduced to methods A, C and generalized cyclic constructions. Where the generalized cyclic construction has not been given by Masuyama (1965) and methods A and C cannot be used generalized cyclic constructions are given in Appendix A2.4. Of the six designs for which no simple generator has been found, three T22, T40 and T45 are ad hoc constructions originally given by Chang, Liu & Liu (1965) and three T77, T85, and T91 are obtained by treatment section (T85) or block-treatment section applied to symmetric BIB designs with $\lambda = 2$. These six designs are readily stored complete. The same methods of construction therefore suffice for both Latin square and triangular designs.

Example 2.7 Clatworthy (1973) derives design T41

(v = 36, r = 7, k = 4, b = 35) from design T54

(v = 45, r = 7, k = 5, b = 35) using a theorem of Shri. kande (1965). T54 is derived from design PG5 which is in turn derived as the dual of PG10; the association schemes for T41 and T54 are unknown. A design equivalent to T54 can be derived by cyclically developing the following initial blocks, in subsets 0-8, 9-17, etc.

0	2	22	23	42
0	3	14	15	38
0	4	29	35	43
9	13	23	25	39
9	12	31	33	43
18	22	28	29	44
0	13	26	30	41

The association scheme is

.	0	9	18	27	32	24	16	8	36
0	.	1	10	19	28	33	25	17	37
9	1	.	2	11	20	29	34	26	38
18	10	2	.	3	12	21	30	35	39
27	19	11	3	.	4	13	27	31	40
32	28	20	12	4	.	5	14	23	41
24	33	29	21	13	5	.	6	15	42
16	25	34	30	22	14	6	.	7	43
8	17	26	35	31	23	15	7	.	44
36	37	38	39	40	41	42	43	44	. .

A design equivalent to T41 may be obtained by omitting treatments 36-44.

2.6 Resolvable block designs

Resolvable block designs have been used extensively in

agricultural variety trials; they are discussed by Cochran and Cox (1957, Chapters 9 and 10). Important resolvable block designs include lattice, rectangular lattice and generalized lattice designs (Patterson and Silvey (1980)). In this section methods are discussed for constructing resolvable $R(v,r,k,b;s)$ block designs with s blocks per superblock. Emphasis is placed upon generalized cyclic constructions. (Resolvable cyclic designs have proved less useful and are not discussed. Quasi-factorial designs, other than lattice designs, derivable using design key operations are also not discussed for their properties are readily determined from those of the related factorial design.) The emphasis is on obtaining efficient designs and no attempt is made to obtain balance or partial balance with very few associate classes.

Two forms of generalized lattice designs are considered, the ϕ series and the α series. The ϕ series contains lattice and rectangular lattice designs and is derived by deleting $s - k \geq 0$ treatments in each block of a (square) lattice design. The α series has a generalized cyclic construction based on two treatment pseudo-factors. For odd values of s the two series overlap. The α series is defined for $k > s$ but the ϕ series is not. We show that the design key generator and generalized cyclic generators (§2.4.2) are adequate for constructing both lattice and generalized lattice designs.

If the off-diagonal elements of the concurrence matrix of a binary design have values $c_1 \leq c_2 \leq \ldots \leq c_p$ we call it a $C(c_1,c_2 \ldots, c_p)$ design. An intuitive procedure when seeking efficient designs (see §2.4.1) is to minimize the variance of the off-diagonal elements. For small values of c_p, this effectively means minimizing c_p and the number of concurrences of order c_p.

Thus a $C(0,1)$ design is likely to be preferred to a $C(0,1,2)$ design if the former exists. Patterson and Williams (1976) show that a necessary, but not sufficient, condition for the existence of a resolvable $C(0,1)$ block design with $1 < r \leq s$ superblocks is $k \leq s$.

For given values v,r,s and k the existence of a $C(0,1)$ in either the ϕ or α series does not necessarily imply the existence of a $C(0,1)$ design in the other.

Example 2.8 For $v = 30$, $s = 6$, $k = 5$, five orthogonal $\cdot$ $\cdot$ rectangles can be generated by cycling modulo 6 from the initial blocks

0	0	0	0	0
0	1	2	5	3
0	2	5	4	1
0	3	1	2	4
0	4	3	1	2

By identifying the rows with one pseudo-factor and the symbols with the other, these rectangles yield a $C(0,1)$ α series design with five replicates. No $C(0,1)$ ϕ series design exists with more than three replicates, when $s = 6$.

The existence of orthogonal Latin rectangles which can neither be generated cyclically nor extended to orthogonal Latin squares implies the existence of yet other $C(0,1)$ generalized lattice designs which could reward investigation, (see §2.6.2).

2.6.1 Lattice designs

In this section some constructions for lattice designs are given. Designs for r superblocks of s^2 treatments can be derived from $r-2$ orthogonal Latin squares of order s but we present methods in which the squares are not explicitly obtained. Methods for constructing orthogonal Latin squares are discussed in Chapter 5.

The standard procedure for constructing lattice designs with r replicates of s^2 treatments (cf Cochran and Cox (1957)) is as follows. Form $r-2$ orthogonal $s \times s$ Latin squares then arrange the $v = s^2$ treatments in an $s \times s$ treatment names array V ; the blocks in the superblocks are determined as follows:

a) Superblock 1: the columns of V

b) Superblock $i+1$: the symbols of the ith square $i = 1 \ldots r-2$

c) Superblock r: the rows of V

When the adjugation operation is available the following procedure is equivalent but more direct. Let A and B be two treatment pseudo-factors with s levels, and let the blocks be formed by the columns of an $s \times s$ array. Then for the first $m-1$ superblocks confound A with the rows of the array and

a) Superblock 1: confound B with columns

b) Superblock $i+1$: form the adjugate of the ith square by interchanging treatments and columns; the symbols of the square determine the level of B .

c) Superblock m: confound B with rows and A with columns.

For some lattice designs more direct methods are available.

i) If s is prime then orthogonal Latin squares are generated from the basic complete array for rows and columns (§2.3) by premultiplying by vectors $(c_i, 1)$ where $1 \leq c_i \leq s-1$. Two squares for which $c_i + c_j = 0$ are adjugates formed by interchanging treatments and columns. For these squares therefore the adjugation operation at step b) is redundant and lattice designs can be derived directly from matrix multiplication (see Example 2.2). The first and $(n+1)$th superblock are given by premultiplying the basic array by matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

This construction extends to designs for which s is not prime (§5.4)

ii) Lattice designs can be constructed from orthogonal cyclic or modified cyclic Latin squares (§5.5) but it is then possible to generate the designs directly.. For cyclic Latin squares (s odd) the construction is identical with that for α series generalized lattice designs (§2.6.2.2). For modified cyclic squares the procedure is illustrated by the following example for $s = 6$.

Example 2.9 Form an $s \times s$ treatment names array and number the treatments as illustrated (cf. the Latin square association scheme §2.5.2)

31	16	11	6	1	21
2	32	17	12	7	22
8	3	33	18	13	23
14	9	4	34	19	24
20	15	10	5	35	25
26	27	28	29	30	36

Select the modified cyclic Latin square

0	2	4	1	x	3
x	1	3	0	2	4
3	x	2	4	1	0
2	4	x	3	0	1
1	3	0	x	4	2
4	0	1	2	3	x

Divide the s^2 treatments into $s+1$ sets of $s-1$ treatments (1...5,6...10 etc.) and one other (36), then the s blocks in each superblock contains $s-1$ generated cyclically and one other. When superblocks are formed in standard order the last block corresponds to

a) Superblock 1: the treatments in the last column

b) Superblock i + 1: the invariant treatment of the ith square (i = 1,..m - 1)

c) Superblock m: the treatments in the last row

The above squares yield the following lattice design where the columns denote blocks:

2	3	4	5	1	21	10	6	7	8	9	1	1	2	3	4	5	26
8	9	10	6	7	22	12	13	14	15	11	2	6	7	8	9	10	27
14	15	11	12	13	23	19	20	16	17	18	3	11	12	13	14	15	28
20	16	17	18	19	24	23	24	25	21	22	4	16	17	18	19	20	29
26	27	28	29	30	25	27	28	29	30	26	5	21	22	23	24	25	30
31	32	33	34	35	36	31	32	33	34	35	36	31	32	33	34	35	36

This design has generalized cyclic construction:

[(01 12 23 34 50 60) mod(1,5) (40 41 42 43 44 ∞) mod(1,1)]

[(14 21 33 42 51 60) mod(1,5) (00 01 02 03 04 ∞) mod(1,1)]

[(00 10 20 30 40 60) mod(1,5) (50 51 52 53 54 ∞) mod(1,1)] .

2.6.2 Generalized lattice designs

Patterson and Silvey (1980) represent generalized lattices as the rows, columns and transversals of an $s \times s$ variety (treatment) names array. For ϕ series designs the method is a straightforward extension of that attributed to Watson by Cochran and Cox (1957), namely that $s - k$ transversals common to $r - 2$ orthogonal Latin squares of order s are deleted. For α series designs the transversals of the squares are cyclically generated diagonals; if the arrays can be completed to form (orthogonal) Latin squares the α design is also a ϕ design. An alternative representation can be in terms of $k \times s$ Latin rectangles for s symbols. Form a $k \times s$ array of variety names and $r - 1$ orthogonal $k \times s$ Latin rectangles. The columns of the variety names array defines the blocks of the first

replicate; the symbols of the ith rectangle define the blocks of the i + 1th replicate (i = 1,2...r - 1). For ϕ designs the Latin rectangles can be extended to Latin squares and the method corresponds to an extension of the method attributed to Shrikande by Cochran and Cox (1957) (see §2.6.2.1) For α designs the Latin rectangles are constructed cyclically from the first column and cannot always be extended to orthogonal Latin squares.

The existence of m orthogonal cyclic Latin squares of order s (Chapter 5) implies that for any $k \leq s$ and $r \leq m+1$ there exists a C(0,1) α design which is also a ϕ design. The methods outlined in §2.6.2.1 may be used to construct ϕ designs for any values of r, s and k if they exist but are recommended only when there is no equivalent α design.

2.6.2.1 Generalized lattice designs- ϕ series

Shrikande's method:

In this method a ϕ (v,r,k,b; s) design is constructed from r + 1 superblocks of a lattice design for s^2 treatments by deletion of all treatments occurring in s - k blocks of one superblock. It is particularly useful if the superblocks of the underlying lattice design are constructed in the order suggested in §2.6.1, i.e. with columns forming the blocks, pseudo-factor A is confounded with rows in r superblocks of the lattice and with columns in the r + 1th. Omitting the last superblock and deleting the last s - k rows of the remaining superblocks causes the last s(s - k) treatments to be deleted and hence avoids the need to renumber treatments or to eliminate empty cells.

Example 2.10 A ϕ(12,3,3,12; 4) design may be constructed by deleting

the last row of the lattice design.

block	a	b	c	d	a	b	c	d	a	b	c	d
	1	2	3	4	1	2	3	4	1	2	3	4
	5	6	7	8	6	5	8	7	7	8	5	6
	9	10	11	12	11	12	9	10	12	11	10	9
	13	14	15	16	16	15	14	13	14	13	16	15

Watson's method:

In this method a $\phi(v,r,k,b;\ s)$ design is constructed from r superblocks of a lattice design for s^2 treatments by deletion of $s-k$ disjoint common transversals of the $r-2$ underlying Latin squares. The method is more general than that of Shrikande when only $r-2$ orthogonal squares of order s exist, the most useful designs being $\phi(30,3,5,18;\ 6)$, $\phi(90,4,9,40;10)$, $\phi(182,4,13,56;14)$. The Shrikande procedure, however, is usually simpler to perform because the Watson procedure requires finding common transversals for the $r-2$ underlying Latin squares. In practice, the Watson procedure must be used when $s = 4m+2$ $(m=1,2...)$ i.e. when the modified cyclic method is the most useful for constructing the underlying Latin squares (§5.5). For these squares the leading diagonal provides a common transversal and may be deleted. A lattice design constructed from modified cyclic Latin squares is given in Example 2.9; deletion of the common transversal causes deletion of the last row of the lattice i.e. the last s treatments. The rectangular lattice may therefore be constructed by omitting the last treatment in each block of the generalized cyclic construction (i.e. treatments 60 and ∞).

2.6.2.2 Generalized lattice designs - α series

An $\alpha(v,r,k,b;\ s)$ design is constructed as follows:

1) Divide the sk treatments into k sets of size s, i.e. $1...s;$ $s+1...2s;$ $...(k-1)s+1\ ...ks;$

2) For each superblock arrange the treatments in a $k \times s$ array, with the ith row a cyclic permutation of the ith set.

3) The columns of the array form the blocks.

The α series designs are essentially generalized cyclic designs for two treatment pseudo-factors A and B with k and s levels respectively. In each superblock the initial block is formed by selecting k treatments no two of which have the same level of A. The remaining blocks are then formed by cycling through the levels of B. When the k elements of each block are ordered by the levels of A then the design is defined by a $k \times r$ array of elements whose jth column contains the levels of B in the initial block of the jth superblock. This array we call the generating array; it corresponds to $f(P_1, P_3)$ in special case iii) of §2.3.

Example 2.11 For $r = 2$, $s = 5$, $k = 4$ the generating array

0	0
0	2
0	1
0	3

yields the design:

block	a	b	c	d	e	a	b	c	d	e
	1	2	3	4	5	1	2	3	4	5
	6	7	8	9	10	8	9	10	6	7
	11	12	13	14	15	12	13	14	15	11
	16	17	18	19	20	19	20	16	17	18 .

Patterson and Williams (1976) point out that if one generating array can be derived from another by a succession of the following operations then the associated designs are isomorphic:

i) addition modulo s of a constant to any row or column,

ii) permutation of rows or columns,

iii) multiplication of all elements by a number coprime to s.

In particular, each α design is isomorphic with one which has first row and column of the generating array identically zero. They also observe that a $\alpha(v,r,k,b;\ s)$ design is a $C(0,1)$ design if

$$\ell_{ij} - \ell_{ij'} \neq \ell_{i'j} - \ell_{i'j'} \qquad \text{(modulo } s\text{)}$$

for all $i \neq i'$, $j \neq j'$, and ℓ_{ij} etc are elements of the generating array.

For each of the following cases there exists an α design which is also a lattice design $(k = s)$ or a ϕ design $(k < s)$. (Most of the results are given by Williams (1975) but all are readily derived from results on cyclic orthogonal Latin squares (Chapter 5)).

1) $r = 2,\ s \geq 2,\ k \leq s$

2) $r = 3,\ s$ odd, $k \leq s$

3) $r = 4,\ s$ odd, $s \neq 0 \bmod 3$ $k \leq s$

4) $r = 4,\ s = 15, 21, 27,\ k \leq s$

5) $r = s,\ s$ prime, $k \leq s$

For $k > s$, some pairs of treatments must occur together in more than one block. Williams (1975) considers the construction of efficient $C(0,1,2)$ designs for $r = 2,3,4$ replicates. He restricts attention to designs with minimum variation among the off-diagonal elements of the concurrence matrix, and, if possible, to partially balanced designs. He then selects the best generator available. We do not discuss this work but note the existence of the extensive tables he prepared.

2.6.3 Dual designs

The block and treatment structures of a resolvable block design may be written (§1.3).

Block $S/B_1/B_2$

Treatment $A_1;A_2 @ R$

where A_1 and A_2 are treatment pseudo-factors, and R is a dummy replication factor; A_2 and B_1 have s levels; A_1 and B_2 have k levels; R and S have r levels. The dual design has block and treatment structures:

$$A_1;A_2/R \quad \text{or} \quad A_1/A_2/R$$

$$S;B_1 @ B_2$$

where B_2 is now a dummy replication factor.

The dual of an α design with generating array G is an α design with generating array H where $h_{ij} = s - g_{ji}$. The dual of a lattice or a ϕ design is not necessarily a ϕ design but if the underlying Latin squares form a complete set and the design is constructed by the method of §2.6.1 with the 'row' superblock omitted or by Shrikande's method as described in §2.6.2 then the dual is a ϕ design. The proof is given in Appendix A2.5.

Example 2.12 The $\phi(12,3,3,12;\ 4)$ design given in Example 2.10 (§2.6.2) is its own dual.

Dual designs are most useful for constructing efficient designs for $b > v$, $k < r$ from designs with $b < v$, $k > r$. Resolvable paired comparison designs ($k = 2$) are the duals of resolvable two-replicate designs and the two classes may be discussed as one.

2.6.4 Resolvable 2-replicate designs

A $R(v,r,k,b;\ s)$ design is associated with a design with $r - 1$ superblocks, each superblock containing a symmetric incomplete block design for s varieties in blocks of k. The association is formed by replacing, in all superblocks other than the first, each treatment by the number of the block in which it occurred in the first

replicate. The association is thus between a $R(sk,r,k,rs;\ s)$ design A (§2.6) and a k-resolvable design B with parameters $(s,k(r-1),k,(r-1)s;s)$. B is the contraction of A and A the expansion of B. A special case of importance is provided when $r=2$ for then B is a symmetric incomplete block design.

For two-replicate resolvable designs Patterson and Williams (1975) show that there is a direct relationship between the efficiency factors of the resolvable design and its contraction or, more precisely, between the efficiency factors of their dual designs. Efficient $R(v,2,k,b;\ s)$ designs may therefore be constructed from efficient symmetric designs for s treatments in blocks of k. Lattice designs are expansions of complete block designs and simple rectangular lattice designs expansions of BIB designs for s treatments in blocks of $s-1$. Symmetric designs with $k>s$ can be formed from $[k/s]$ complete blocks together with an incomplete block size $k' = k \bmod s$. Williams, Patterson and John (1977) give a list of efficient symmetric designs and alternative cyclic designs.

The expansion operation is valuable in design construction and is discussed further in §7.2.4. The relationship between the efficiency factors extends to two replicate resolvable row and column designs (Appendix A8.3) but not to three-replicate resolvable designs. It is likely that the expansion operation can still be used with $r>2$ to construct designs of high efficiency but the advantages of working with the contracted design are much less pronounced than for $r=2$.

2.6.5 Availability of designs

The usefulness of cyclic and generalized cyclic designs is dependent on the availability of algorithms for generating initial

blocks yielding efficient designs or on the availability of tables or computer files of suitable initial blocks. Little work appears to have been done so far on the construction of algorithms. To date the following sets of tables have been reported:

i) Cyclic designs: John et al (1972), John (1981).

ii) Generalized lattice-α series: Williams (1975), Patterson, Williams and Hunter (1978)

iii) Generalized lattice-ϕ series: None.

iv) Resolvable two replicate designs: Williams, Patterson and John (1977).

2.7 Resolvable row and column designs

Resolvable row and column designs permit more control of variability than resolvable block designs and can prove useful if plots are nearly square. The most widely used form - lattice squares - are described by Cochran and Cox (1957, Chapter 12).

Construction methods for resolvable block designs are not necessarily useful for resolvable row and column designs. For example, the methods given for lattice designs in §2.6.1 and α series designs in §2.6.2 cause the same effects to be confounded with rows in each superblock. Simple alternative constructions exist, however, for lattice square designs with s^2 treatments; Kempthorne (1952, Chapter 24) gives some examples.

Let there exist m orthogonal squares of order s, then the rows and columns of the treatment names array V and the letters of the m squares define $m+2$ categories for determining the rows and columns of the lattice design. The columns and the rows of the squares can be determined as follows:

$m = 2\ell$	Superblock	1	2	3	....	$\ell+1$
	Columns	C	1	2		ℓ
	Rows	R	$\ell+1$	$\ell+2$		m

where $1,2\ldots\ \ldots m$ denote the letters of the appropriate square and C and R denote the rows and columns of the treatment names array V. For $\ell+1 < r \leq 2(\ell+1)$ rows and columns, can be interchanged if necessary.

$m = 2\ell-1$	Superblock	1	2	3	....	$\ell+1$	$\ell+2$	....	$m+1$	$m+2$
		C	1	2		ℓ	$\ell+1$		m	R
		R	$\ell+1$	$\ell+2$		C	1		$\ell-1$	ℓ

If the ith orthogonal latin square $(1 \leq i \leq m)$ can be constructed by pre-multiplying the basic complete array (§2.3) by matrix $(E_i\ F_i)$ then the above sequences can be constructed by premultiplying the basic array by matrices

m even $\begin{pmatrix} I & O \\ O & I \end{pmatrix} \begin{pmatrix} E_1 & F_1 \\ E_{\ell+1} & F_{\ell+1} \end{pmatrix} \ldots \begin{pmatrix} E_\ell & F_\ell \\ E_m & F_m \end{pmatrix}$

m odd $\begin{pmatrix} I & O \\ O & I \end{pmatrix} \begin{pmatrix} E_1 & F_1 \\ E_{\ell+1} & F_{\ell+1} \end{pmatrix} \ldots \begin{pmatrix} O & I \\ E_\ell & F_\ell \end{pmatrix}$

Example 2.13 For $s = 3$ two orthogonal latin squares can be constructed by pre-multiplying the basic array by matrices (2, 1) and (1, 1) respectively. A balanced set of two super blocks is then given by pre-multiplying the basic array by matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

i.e.

1	2	3	1	5	9
4	5	6	8	3	4
7	8	9	6	7	2

Before considering the construction of more general row and column

designs we note that when s is prime a balanced lattice can be constructed through the use of two treatment pseudo-factors and pre-multiplication of the basic complete array by matrices of the form

$$C_i = \begin{pmatrix} d_i e_i + 1 & e_i \\ d_i & 1 \end{pmatrix}$$

for suitably chosen d_i and e_i. Now,

$$C_i = \begin{pmatrix} 1 & e_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ d_i & 1 \end{pmatrix} = U_i L_i$$

and thus each superblock the lattice square can be constructed in two stages equivalent to premultiplying the basic array by L_i and then by U_i. This is equivalent to forming the superblock in two stages:

i) For each level of the first factor cycle the levels of the second factor

ii) For each level of the second factor cycle the levels of the first factor.

Example 2.14 The second superblock in Example 2.13 is formed as follows:

00	01	02	→ 00	01	02	→ 00	11	22 → 1 5 9
10	11	12	11	12	10	21	02	10 8 3 4
20	21	22	22	20	21	12	20	01 6 7 2

The following procedure provides a generalization for resolvable row and column designs with r c treatments.

1) Form an $r \times c$ treatment names array with levels of treatment pseudo-factor A defining the rows and the levels of factor B the columns. For the ith superblock, i = 2...r:

2) For each level j of factor A add (Modulo c) d_{ij} to each level of B

then 3) for each level k of factor B add (Modulo r) e_{ik} to each level of A

The procedure defined is therefore essentially two α design constructions applied by cycling the levels of factors A and B respectively. More general constructions can be defined by allowing more complex permutations of the levels of pseudo-factors A and B. The construction of efficient row and column designs by the method suggested clearly requires investigation with the aid of a computer.

The contraction and expansion operations of Patterson and Williams (1976) (§2.6.4) may be extended to two-replicate row and column designs. In Appendix A8.3 we show that the relationship between the efficiency factors of the resolvable design and its contraction can be suitably modified. It is probable therefore that efficient row and column designs for two replicates may be formed from modifying the approach of Williams, Patterson and John (1978). This remains to be done.

Appendix A2.1

Construction of unreduced BIB designs for k = 2

A simple construction is presented for 2-resolvable (v odd) and resolvable (v even) unreduced BIB designs for $k = 2$. Consider the case for $v = 2m+1$, the m pairs $(j, v-j)$, $j = 1,2 \ldots m$ contain all non-zero differences modulo v. These m pairs may be used as initial blocks to cyclically generate the $m(2m+1)$ blocks:

(1, v-1)	(2, v-2)	(3, v-3)	....	(m, v-m)
(2, 0)	(3, v-1)	(4, v-2)	....	(m+1, v-m-1)
. . .	. . . . .	. . . . .	. . . .	. . . . .
(0, v-2)	(1, v-3)	(2, v-4)	....	(m-1, v-m+1)

Each column forms a complete replicate and the design can be arranged as an extended Youden square. The ith treatment ($i = 0,1 \ldots, v-1$) is missing from the ith row, and thus treatments are balanced for complete rows.

For $v = 2m+2$ construct the array for $v' = 2m+1$ as above and supplement the array with a column containing the treatment pair (i,∞) ($i = 0,1 \ldots, v'-1$). All possible pairs occur in the resulting array and each row of the array forms a complete replicate.

Appendix A2.2

Generalized cyclic constructions for some BIB designs

Some constructions for BIB designs are presented which may be used as alternatives to those in Tables XVII and XIX of Fisher and Yates (1963). Those marked * are preferred.

*Design 7: [(01 11 31)(51 12 22)(41 32 52)(21 02 42)(∞ 61 62)] mod (7,1)

Design 20: See Preece (1972) for a generalized cyclic construction not yielding an extended Youden square.

Design 25: Raghavarao (1971):

(01 12 23)(01 22 43)(01 32 63)(01 42 13)(01 52 33)

(01 62 53)(01 02 03)(01 11 31)(02 12 32)(03 13 33) mod (7,1)

Design 27: See Fisher and Yates (1963, p.90) for a generalized cyclic solution not yielding an extended Youden square.

*Design 34: [(0 1 3 4 5 9)(2 6 7 8 10 ∞)] mod 11

*Design (14, 13, 7, 26; 6) Takeuchi (1962):

(0 1 3 4 9 10 12)(∞ 1 3 4 9 10 12) mod 13

*Design 48: (0 1 18 30)(0 2 13 16)(0 5 9 15) mod 37

*Design 49: (001 002 003 004 012 022 104 111 123 204 213 221) mod (3, 3, 5)

*E(3,3): 1 Construct 13 superblocks using the distinct powers of design key matrix

$$\begin{bmatrix} 0 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

*P(3,3): 1 Kageyama (1972) [(∞ 00 01 02)

(10 120 81 51) (40 90 61 71) (30 100 111 21)

(11 121 82 52) (41 91 62 72) (31 101 112 22)

(12 122 80 50) (42 92 60 70) (32 102 110 20)] mod (13,1)

Appendix A2.3

Constructions for Latin square PBIB designs

In this appendix the 146 PBIB(2) design with Latin square association scheme listed by Clatworthy (1973) are catalogued by method of construction; where Clatworthy reports several constructions only one is given. Six categories are used. Methods A, B, C, D, E and 'others'; methods A to D are defined in §2.6.2 and method E is a geometrical E(2,n) construction. Only 19 designs are not constructed by methods A or C and generalized cyclic constructions are presented for these in the second table.

Constructions

Method A (18 designs)

Nos.(LS) 1,2,3,4,5,6,16,17,23,24,25,45,46,47,48,67,70,71

Method B (4 designs)

Nos.(LS) 72,98,99,135

Method C (109 designs)

Nos.(LS) 7,...,15,19,21,27,28,29,...,37,39,...,44,50,51,...,66,69,
73,74,...,81,84,...,97,102,...,115,119,...,133,137,...,146.

Method D (7 designs)

Nos.(LS) 26,49,83,101,116,117,136

Method E (4 designs)

Nos.(LS) 18,68,82,134

Others (4 designs)

Nos.(LS) 20,22,38,100,118

Generalized cyclic alternatives to methods B, D, E and others

B.LS72	(00 01 02 10 11 12) mod (3, -)
	(00 10 20 01 11 21) mod (-, 3)
LS98*	[(00 01 02 03 10 11 12 13)(20 21 22 23 30 31 32 33)] mod (3, -)
	[(02 10 20 30 01 11 21 31)(02 12 22 32 03 13 23 33)] mod (-, 3)
LS99	Merge suggested blocks (Clatworthy, 1973) to LS98
LS135	(00 01 02 03 04 10 11 12 13 14) mod (5, -)
	(00 01 02 03 04 20 21 22 23 24) mod (5, -)
	(00 10 20 30 40 01 11 21 31 41) mod (-, 5)
	(00 10 20 30 40 02 12 22 32 42) mod (-, 5)
D.LS26	(01 02 10 20) mod (3, 3)
LS49	(00 01 02 10 20) mod (3, 3)
LS83	(00 01 02 03 10 20 30) mod (4, 4)
LS101	(01 02 03 04 10 20 30 40) mod (5, 5)
LS116	(11 12 13 21 22 23 31 32 33) mod (4, 4)
LS117	(00 01 02 03 04 10 20 30 40) mod (5, 5)
LS136	(01 02 03 04 05 10 20 30 40 50) mod (6, 6)
E.LS18	(00 12 31)(23 00 32)(21 13 00)(02 11 23) mod (4, -)
LS68,82,134	see Clatworthy (1973)
Others LS20	Use design LS 17
LS22	(00 01 40) (00 03 20) mod (5, 5)
LS38	Use design LS 42
LS100	Use design LS 99
LS118	See Clatworthy (1973)

* Level 3 is not affected by cycle.

Appendix A2.4

Constructions for triangular PBIB designs

A method for constructing each of the 100 triangular PBIB designs given by Clatworthy (1973) is given. Methods A-E are as described in Appendix A2.3. Two other methods are useful:-

F) Form blocks of $m(m-1)/2$ treatments $(m < n)$ which are related by a (sub) triangular association scheme.

G) Form the dual of a BIB design. (The fundamental designs may be constructed by this method.)

The second list contains generalized cyclic constructions to complement those given by Masuyama (1965)

Constructions

Method A (14 designs)

Nos.(T) 1,5,7,10,21,23*,25*,38,39,42,43,53,55,56

Method C (46 designs)

Nos.(T) 3,4,11,15,17,18,19,29-32,34-37,46,47,49-52,58,59,63,64, 66-69,73-76,79-82,87-90,93,97-100

Method D (8 designs)

Nos.(T) 9,33,60,61,71,84,94,95

Method F (5 designs)

Nos.(T) 14,20,24,26,27,70 (also 23,25)

Method G (13 designs including 7 fundamental designs)

Nos.(T) 28,44,48,57,62,65,72,78,83,85,86,92,96

Generalized Cyclic (8 designs)

Nos.(T) 2,6,8,12,13,16,41,54

Other (6 designs)

Nos.(T) 22,40,45,77,91

*These solutions are not isomorphic to those given by Clatworthy

Cyclic alternatives to methods D,F and G not given by Masuyama (1965)

T8 (0 6) (0 9) (0 10) (0 8) (0 7)

(1 4) (1 10) (1 11) (1 9) (1 5)

(2 5) (2 8) (2 12) (2 7) (2 3) (Increment, I = 3, V = 21)

T12 (Preferred to Masuyama's solution)

(0 1 5) (0 2 6) (3 0 5) (7 9 0) (I = 2, V = 10)

T26 (0 1 5) (0 6 2) (0 46 47)

(1 3 11) (1 12 4) (1 42 44)

(2 19 42) (2 50 53) (2 3 15)

(3 42 23) (3 24 46) (3 34 21)

(4 50 29) (4 0 8) (4 20 3) (I = 5, V = 55)

T27 (1 2 7) (1 3 8) (1 4 9) (1 5 10) (4 5 25)

(2 3 13) (2 4 14) (2 5 15) (3 4 19) (3 5 20)

(1 41 11) (2 41 16) (2 46 17) (3 46 22) (5 51 33) (I = 6,

(0 6 1) (0 12 2) (0 18 3) (0 24 4) (0 30 5) V = 66)

T41 See example (§2.5.2)

T54 See example (§2.5.2)

T70 (0 1 2 3 4 6) (0 1 14 3 5 7) (0 2 16 4 17 10)

(0 1 16 3 17 8) (0 2 14 4 5 9) (I = 3, V = 21)

Other designs

Eleven designs are not covered by Masuyama (1965) or the above list. T60, T71, T83, T84, and T 92 are the complements of designs T33, T9, T62, T61 and T48 respectively and their generators are readily derived. No generalized cyclic constructions have been found for designs T22, T40, T45, T77, T85, T91.

Appendix A2.5

Dual of ϕ series designs

In this appendix we show that if s is a prime power then ϕ series designs may be constructed whose duals are also ϕ series designs.

If s is a prime-power then there exists a field of order s. Denote the elements of the field $\alpha_0, \alpha_1 \cdots \alpha_{s-1}$ where α_0 denotes the zero element.

A lattice design with s superblocks can be represented by block and treatment structures

$$S/B/P$$

$$A_2; A_1 @ S'$$

where A_1 and A_2 are treatment pseudo-factors and S' is a dummy replication factor. Let the treatments be allocated by the rule

$$S' = S$$

$$A_1 = SP + B$$

$$A_2 = P$$

that is if S, B and P have levels α_i, α_j and α_k then S', A_1 and A_2 have levels $\alpha_i, \alpha_i\alpha_k + \alpha_j$ and α_k respectively. The resulting design is an s replicate lattice design (see Example 2.2 §2.3). The dual design has block and treatment structure

$$A_2/A_1/S'$$

$$S;B @ P$$

with relationship

$$P = A_2$$

$$B = A_1 - PS = A_1 - A_2 S'$$

$$S = S'$$

which is also a s-replicate lattice design.

If levels of S are deleted from the original design (i.e. superblocks are deleted) then in the dual design levels are deleted from the plot factor S' . If levels of P are deleted in the original design (i.e. certain treatments are deleted) then in the dual design levels of the superblock factor A_2 are deleted. The s + 1th superblock of the original and dual designs are defined by levels of A_2 and S respectively, so deletion of levels of these factors in other superblocks corresponds to Shrikande's method of constructing ϕ designs.

Example The ϕ (25, 2, 4, 10; 5) design

a	b	c	d	e
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

a	b	c	d	e
1	2	3	4	5
7	8	9	10	6
13	14	15	11	12
19	20	16	17	18
25	21	22	23	24

has dual ϕ (10, 4, 2, 25; 5) design

1	2	3	4	5
6	7	8	9	10

1	2	3	4	5
10	6	7	8	9

1	2	3	4	5
9	10	6	7	8

1	2	3	4	5
8	9	10	6	7

1	2	3	4	5
7	8	9	10	6

CHAPTER 3

CONFOUNDED AND FRACTIONAL DESIGNS FOR p^n FACTORIAL EXPERIMENTS

3.1 Summary

In this chapter generators for constructing confounded or fractional prime power p^n factorial experiments are described. Attention is restricted to designs with simple block structure and with fractions and confounding determined by selected defining contrasts. Relevant literature is reviewed in §3.2. In §3.3 three related generators due to Das (1964), John and Dean (1975) and Patterson (1976) are described and compared; their use with asymmetric factorial experiments is deferred to Chapter 4. The most general of the three, the design key method, is discussed in more detail and applications are given to partially confounded designs and quasi-Latin squares.

The construction methods described in §§3.4 and 3.5 cover somewhat complementary circumstances. Those of §3.4 are suitable when little is known about which factors will prove most important or when their effects are likely to be of the same order. A 'best' system of defining contrasts is defined and methods for obtaining such a system are described. Compact matrices are presented from which a wide variety of designs comparable with those in published tables, can be formed. The methods of §3.5 are suitable when it is possible to specify which factors are likely to have sizeable effects and for which effects estimates are required. An algorithm is given for selecting an acceptable system of defining contrasts for a p^n factorial experiment in p^m blocks of p^{n-m} units to

meet the user's requirements. Also given are procedures for applying the algorithm to more general problems or for making it more efficient in problems which are frequently encountered.

The topics, discussed in Chapters 3 and 4 are related and some methods for constructing other fractions of p^n factorial experiments are deferred to Chapter 4.

3.2 Review of the literature

The principles and theory of confounded and fractional p^n factorial experiments were established in the early work of Yates (1933, 1937a), Bose and Kishen (1940), Finney (1945) and Bose (1947). The standard theory is based on finite geometries and related Galois fields; a good description is given by Kempthorne (1952).

Confounded designs are regularly used in agricultural experiments. Fractional designs have been widely used in industrial experimentation, sometimes in their own right (see Daniel (1976)) but often as a basis for response surface designs (Box and Wilson (1951)). The most common symmetrical factorial experiments are those with $p = 2$ or 3. Designs with $p = 4$ can be treated as a special case of $p = 2$ by substituting two two-level pseudo-factors for each four-level factor; designs with $p \geq 5$ are uncommon. The p^{n-m} fractional experiment in p^r blocks of p^{n-m-r} units where $p = 2$ or 3 is of particular importance. Extensive tables of two and three-level factorial experiments of this type are given in USA National Bureau of Standards (NBS) (1957, 1959).

Cochran and Cox (1957, chapters 6 to 8) present a range of useful plans, for fractional, confounded and partially confounded experiments.

We follow Cox (1957, p. 259) and apply the term defining contrast to both fractional and confounded factorial experiments. For both types of experiment the set formed from the mean effect I and all $p-1$ ways of writing each of the $(p^m-1)/(p-1)$ defining contrasts is a group of order p^m ; similarly the treatments in the principal block form a group of order p^{n-m} . The defining contrasts can be generated from a suitable subset of size m and the treatments in the principal block can be generated from a subset of size $n-m$. The subset of m defining contrasts are called <u>defining contrast generators</u> and the subset of $n-m$ treatments the <u>treatment generators</u>. Given one set the other can be derived easily. Box and Hunter (1961) describe a construction procedure based on defining contrast generators while Das (1964) describes an otherwise equivalent scheme based on treatment generators. John (1973) presents a generalized cyclic construction, based on treatment generators, useful for both symmetric and asymmetric factorial experiments; a special form of this generator is used by John and Dean (1975) for constructing single replicate symmetric (not necessarily p^n) experiments. Patterson (1965, 1976) and Patterson and Bailey (1978) describe a method which links the various approaches via a design key matrix. The method is not only compact but is readily adopted to a wide range of designs with simple block structure. Other authors (e.g. Finney (1960, §5.10)) have used equivalent procedures for relating defining contrast and treatment generators.

The number of factors in a defining contrast is sometimes called its <u>word length</u>. The <u>resolution</u> of a design is determined by the shortest word length. A design of resolution III (Box and Hunter, 1961) has a minimum word length of three and in the absence of

interactions allows main effects to be estimated. Designs of resolution V , which allow two factor interactions to be estimated in the absence of higher order interactions, are useful for constructing second-order response surface designs. Many designs of resolution V are given in the NBS (1957, 1959) tables. Designs of high resolution commonly require many units; Addelman (1962a) gives a list of compromise designs requiring fewer units but in which certain two-factor interactions are deemed negligible. Greenfield (1976) adopts a different approach of constructing a fractional design to meet specified requirements. Franklin and Bailey (1977) extend the algorithm so that it can be used for any fractional or confounded 2^n design.

3.3 Design generators for p^n factorial experiments

In this section three closely related methods for constructing p^n factorial experiments are described. The methods, due to Das (1964), John and Dean (1975) and Patterson (1976), are called respectively:

a) Das's method.

b) The generalized cyclic method.

c) The design key method.

Das's method is useful for simple confounded or fractional p^n experiments but is not readily generalized. The generalized cyclic method is intended for single replicate designs; it is readily adopted to asymmetric factorials but not to designs with complicated block structure. The design key method is the most general of the three and is compactly expressed.

Preliminary remark. Each treatment factor is regarded as having p levels, pseudo-factors being substituted if necessary. For n

treatment factors each treatment may be represented by a vector $\underline{q} = (q_1, q_2, \ldots q_n)^T$ where $0 \leq q_i < p$ $(i = 1 \ldots n)$. A defining contrast may be represented by a vector $\underline{x} = (x_1, x_2, \ldots x_n)^T$ where $0 \leq x_i < p$ $(i = 1 \ldots n)$. Thus, for example, when $n = 4$, $p = 3$ the effect AB^2D is presented by the vector $(1,2,0,1)^T$. In this notation if $\underline{x}$ is a defining contrast and $\underline{q}$ is a treatment in the principal block then

$$\underline{q}^T\underline{x} = 0 \qquad 1)$$

where $\underline{q}^T\underline{x} = \Sigma x_i q_i \bmod p$. If Q^T is an $(n-m) \times n$ matrix of treatment generators for the principal block and X is an $n \times m$ matrix of defining contrast generators then

$$Q^T X = 0 \qquad 2)$$

Das's method: For a p^n factorial experiment confounded in blocks of p^{n-m} units, $n-m$ treatment factors $B_1, B_2 \ldots B_{n-m}$ can be found so that all combinations of levels of these factors occur in each block. We call the design for $n-m$ factors the basic design and the factors B_i the basic factors. The remaining m factors are called added factors and denoted A_i . The generators of the basic design may be represented by the rows of a unit matrix of order $n-m$, the jth column relating to factor B_j . The introduction to the basic design of an added factor cannot increase the number of distinct treatment combinations in a block so the levels of the new factor must be identified with a combination of levels of the basic factors. This combination can be represented by vector of length $n-m$ with elements modulo p . There are m such vectors, one for each added factor, which may be represented as the columns of an $(n-m) \times m$ matrix C . The $n-m$ generators of the treatments in the principal block are then represented by the rows of the $(n-m) \times n$

matrix

$$Q^T = (C\ I) \qquad 3)$$

where the first m columns represent the added factors and the last $n-m$ columns the basic factors. If the jth added factor is identified with $\underline{c}_j$ the jth column of C then the interaction E is a defining contrast where

$$E = B_1^{d_1} B_2^{d_2} \cdots B_{n-m}^{d_{n-m}} \cdot A_j$$

and $d_i = p - c_{ij} \bmod p$ and c_{ij} is the ith element of $\underline{c}_j$. Indeed, the defining contrast generators are given by the columns of the matrix

$$X = \begin{pmatrix} I \\ -C \end{pmatrix} \qquad 4)$$

where $-C = pJ - C \bmod p$.

Example 3.1 If in a 3^5 design in blocks of 9 units the treatments in the principal block are generated by the rows of Q^T

A_1	A_2	A_3	B_1	B_2
1	1	2	1	0
1	2	1	0	1

Then the defining contrasts are generated by the columns of X :

A_1	1	0	0
A_2	0	1	0
A_3	0	0	1
B_1	2	2	1
B_2	2	1	2

Das (1964) considers adding columns to matrix C in 3) while Box and Hunter (1961) consider adding columns to matrix -C in 4). The problems are clearly equivalent. Das notes that if no two columns of C are identical and each contains two or more non-zero

elements the designs must have at least resolution III. If for 2^n fractional experiments all the columns of C also contain an odd number (≥ 3) of non-zero elements the design has resolution IV. (These observations are used in §3.4.1.)

Generalized cyclic method: In this method $n-m$ linearly independent rows are chosen to form the rows of the matrix Q^T. These then generate the principal block and m further rows are chosen as block generators to form the full single replicate design. The defining contrast generators are given by the columns of any matrix X of full rank m for which $Q^T X = O$.

With suitable ordering of the factors, if necessary, it is always possible to find a set of generators expressed in the form given by Das (Franklin and Bailey, 1977). The two methods are thus equivalent for p^n factorial experiments. The generalized cyclic method, however, is suitable for use with symmetric, non-prime power factorial experiments and is readily extended to asymmetric factorial experiments.

The design key method: For a factorial experiment with p^k units and p^n treatments Patterson and Bailey (1978) define k plot factors and n treatment factors each with p levels. A block factor with p^u levels is uniquely identified with u plot factors. Any unit can be represented by a $k \times 1$ vector $\underline{r} = (r_1, \ldots, r_k)^T$ where $0 \leq r_j < p$ is the level of the jth plot factor. Treatment and plot effects $\underline{x}$ and $\underline{y}$ are defined as the set of contrasts with $p-1$ degrees of freedom between units with different values mod p of $\underline{q}^T\underline{x}$ and $\underline{r}^T\underline{y}$ respectively. By identifying n plot effects $\underline{y}$ with the n treatment main effects the confounding scheme is determined. The specification of the plot aliases of the treatment main-effects is called the design key.

For single replicate p^n designs the relationship between plot and treatment effects can be expressed in the form:

$$\underline{y} = K\underline{x}$$

$$\underline{x} = K^{-1}\underline{y} \qquad 5)$$

where K, the design key matrix, (or, simply the key matrix) has order $n \times n$ with elements modulo p.

Now, $$\underline{x}^T\underline{q} = \underline{y}^T\underline{r} = \underline{x}^T K^T\underline{r}$$

so $$\underline{q} = K^T\underline{r} . \qquad 6)$$

and hence the treatment allocated to each unit is determined.

Example 3.2 A single replicate design is required for 2^6 treatments in four blocks of 16 units with ACDE, BCDF and ABEF confounded with blocks. Denote the plot effects $P_1, P_2, \ldots P_6$ then

$$ACDE = P_1$$

$$BCDF = P_2$$

$$C,D,E,F = P_3,P_4,P_5,P_6$$

so $$A = P_1P_3P_4P_5$$

$$B = P_2P_3P_4P_6$$

and $$K = K^{-1} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 1&1&1&0&0&0 \\ 1&1&0&1&0&0 \\ 1&0&0&0&1&0 \\ 0&1&0&0&0&1 \end{bmatrix}$$

The treatment applied to plot $(0\ 0\ 0\ 0\ 0\ 1)^T$ is $(0\ 1\ 0\ 0\ 0\ 1)^T$ = bf.

The key matrix K is related to matrix Q^T of Das's method; the rows of K corresponding to the nested block factor generate the treatments in the principal block. The defining contrasts can be determined from the columns of K^{-1}.

As described so far the three methods are equivalent but the design key method is defined more generally and allows for i) any simple block structure ii) fractional designs, iii) pseudo-factors and iv) addition of a constant vector to all treatments. We therefore adopt this procedure for the remainder of the chapter. The use of generalized cyclic and design key methods for asymmetric factorial experiments is discussed in Chapter 4.

3.3.1 Further comments on the design key method for p^n factorial experiments

Geometrical interpretation: The general form of the design key generator for a p^{n-m} fraction can be written as

$$\underline{t} = \underline{t}_0 + K^T\underline{p} \qquad 1)$$

where $\underline{t}$ is the treatment vector, p the plot vector, K the $(n-m) \times n$ key matrix and $\underline{t}_0$ is an arbitrary (base) vector determining the particular fraction. A set of $n-m$ treatment generators may be produced by replacing the plot vector $\underline{p}$ successively by the columns $\underline{e}_j$ of a unit matrix yielding

$$\underline{t}_j = \underline{t}_0 + K^T\underline{e}_j \quad . \qquad 2)$$

If R is the $m \times n$ matrix of whose rows generate the defining contrasts then

$$R\underline{t}_j = R\underline{t}_0 = \underline{c} \quad . \qquad 3)$$

Equation 3) represents a linear $n-m$ flat in $EG(n,p)$ and the solutions represent the points in the flat. R has full rank and therefore a right inverse R^r so for any vector $\underline{c}$ there exists a vector $\underline{t}_0$ such that $\underline{t}_0 = R^r\underline{c}$ and therefore 1) is the general solution to equation 3).

Dummy factors: In its general formulation the key matrix is not necessarily square. There is usually little difficulty

in deriving a 'generalized inverse' of the key matrix by hand but a convenient approach for automatic procedures is provided by the use of dummy factors.

Dummy plot factors I_1, I_2 ... are required for fractional designs and dummy treatment factors Z_1, Z_2 ... for multiple replicate designs (cf. dummy replication factors). Extra rows or columns are then added appropriately to the key matrix to make it non-singular, and the matrix inverted to yield the defining contrasts (the dummy factors being treated as factors with p levels). For a fractional design any treatment effect confounded (aliased) with a generalized interaction containing dummy factor I_j is a defining contrast for the fraction. For multiple replicate designs treatment effects containing any Z_i contribute to error.

Uniqueness: A design key is usually not unique in the sense that experimental plans derived from different design keys must be different. When constructing standard key matrices it is useful to know which matrices are equivalent under valid randomization procedures and to present the matrix in some form regarded as basic. In Appendix A3.1 we show that an equivalent design is obtained if the key matrix is left-multiplied by a block upper triangular matrix

$$D = (\delta_{ij}\, D_{ij})$$

where δ_{ij} is the (j,i) element of the lower triangular nesting matrix M (§1.3) and D_{ii} is non-singular of dimension determined by the ith block factor.

When the key matrix is non-singular the treatment factors, and hence the columns, may be reordered to make the block matrices K_{ii} on the leading diagonals of the key matrix non-singular. By suitable choice of D the block matrices K_{ii} may be reduced to unit matrices. This provides a representation for key matrices which could be regarded as a standard form.

Example 3.3 By suitable reordering of treatments the key matrix for a single replicate confounded design can be represented in the form

$$\begin{bmatrix} I & O \\ C & I \end{bmatrix}$$

See also Example 3.2 and compare with Das's method.

Varying block and treatment structures: A key matrix identifies treatment factors with plot factors, the combination of the block factors to yield the desired block structure is a separate stage. The design key method is therefore available for any simple block structure and one key matrix is often useful for several distinct block structures, e.g. a key matrix for generating a lattice square design is equally useful for generating a lattice design where there is one block stratum fewer (c.f. §2.7). The use of design key matrices to yield multiple designs reduces the number of standard matrices that require to be stored.

A design key is readily adapted to adding plot factors or treatment factors or both merely by identifying new with existing factors. The updating process corresponds with the addition of extra rows or columns to the design key matrices and is implicit in the standard matrices of §3.4.

3.3.2 Partial confounding and Quasi-Latin Squares

The construction of partial confounding schemes for p^n factorial experiments of practical size causes no real difficulty and for most cases suitable schemes have been presented by Cochran and Cox (1957, Chapter 6). Often, the schemes may be produced by use of the same key matrix for all superblocks but with the treatment factors permuted.

Example 3.4 A balanced 2^6 design in 20 blocks of 16 units is obtained by confounding ABCD, ADEI, BCEI in the first superblock and cyclically permuting all but factor I in the remainder.

For designs with this form of partial confounding a slight generalization of the design key method allowing cyclic permutation of the columns of the key matrix could be useful. This could be achieved, for example, by generating the ith replicate through a key matrix of the form $K_i = K E^i$ where E is a suitable permutation matrix.

The design key method can be useful also in the construction of quasi-Latin squares and rectangles (Cochran and Cox (1957, Chapter 8)). Consider the application of p^n treatments to a rectangle of dimensions $p^r \times p^c$ where rows and columns form blocks. Let $u = r + c - n$ then the design is a fractional, single or multiple replicate design depending on whether u is less than, equal or greater than zero. When $u \leq 0$ there is no difficulty in constructing the square; it is necessary only to choose r defining contrast generators for the rows and c for the columns such that the direct product group contains all effects in the fraction. For complex problems the methods of §3.5 may be used.

For designs with $u > 0$ a complete replicate can be contained in p^{n-c} rows or p^{n-r} columns. There is a choice of useful block structures. In particular, the resolvable designs associated with lattice squares are available; these are discussed in §2.7 and not considered here. Useful block structures for quasi-Latin squares are then

a) (R1/R2) * (C1/C2)

or b) (R1*C1) / (R2*C2)

where R1 and C1 have p^u levels and R2 and C2 have p^{n-c} and

p^{n-r} levels respectively. The choice of block structure belongs to the experimenter but those given above illustrate two features: a) ignoring the row or column category yields a (partially) confounded design and b) for each combination of the levels of factors R1 and C1 the design is a p^{-u} fractional quasi-Latin rectangle. The defining contrasts of the fraction are the same for all combinations of R1 and C1 but each coset occurs p^{u} times.

For any square the effects confounded with rows and columns must be compatible with each other and with the defining contrasts of the fraction. If [D] denotes the defining contrast group of the fraction, [R] and [C] the effects confounded with rows and columns for some level of R1 and C1 then [D], [R] and [C] must be disjoint (apart from the mean effect I) and no effect in [C] may be representable as an interaction of effects in [D] and [R] etc. Furthermore, if {R} and {C} denote the collection of effects confounded with rows and columns then {R}, {C} and [D] must be disjoint and no effect in [D] may be representable as the interaction between an effect in {R} and an effect in {C} . The full quasi-Latin rectangle may be constructed as follows:

1) select the confounded effects of the rows and columns compatible with each other;
2) select the defining contrasts of the fraction compatible with the confounded effects;
3) for the (R1 = i, R2 = j) fraction form the (principal) rectangle determined by the fraction and confounded effects.
4) adjust the (i,j) principal rectangle by the appropriate coset of the fraction.

(Minor simplification can be made to stage 3 if two or more superblocks contain the same confounding scheme.)

Example 3.5 An 8×8 quasi-Latin square is required for two replicates of a 2^5 factorial design with

ACE, BCD, ABDE; ACD, BDE, ABCE

confounded with columns and

ABC, ADE, BCDE; ABD, BCE, ACDE

confounded with rows. For this design u=1 and r=c=3.
The design presented by Cochran and Cox (1957, Plan 8.8) may be generated (with some reordering of rows and columns) as follows.
Restrict in turn to

R1=0, C1=0; R1=0, C1=1; R1=1, C1=0; R1=1, C1=1

Generate the treatments from the design key matrices

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

and apply base vectors $(1\ 1\ 0\ 1\ 0)^T$ and $(1\ 1\ 1\ 0\ 0)^T$ to the second and third squares respectively.

There are relatively few quasi-Latin squares of interest and most are small; they are easily stored complete.

3.4 Standard design key matrices for two and three-level factorial experiments

Often when fractional factorial experiments are to be constructed little information is available to indicate which interactions will prove important. Designs with good all-round properties are then

desired. The two manuals of fractional designs with confounding produced by the USA National Bureau of Standards (NBS) (1957, 1959) contain many suitable designs but can be difficult to use and are liable to transcription error.

Daniel (1976) discussing published tables of fractional 2^n and 3^n designs expresses the need for '... a clearly printed, lexical, computer checked printout of [the NBS (1957, 1959)] plans'. No algorithm for doing this appears to have been published. In this section we describe a simple method of generating many useful designs and thus effectively storing a manual within the computer.

Consider a p^{n-m} fractional design with block structure $I/S_1/S_2$ where I is a dummy block factor corresponding to the fraction and I, S_1 and S_2 have p^m, p^r and p^{n-m-r} levels respectively. The discussion in §3.3.1 and Appendix A3.1 demonstrates that for some ordering of the treatment factors the design key matrix can be expressed in lower triangular form

$$\begin{vmatrix} L_{11} & 0 & 0 \\ A_{21} & L_{22} & 0 \\ A_{31} & A_{32} & L_{33} \end{vmatrix} \qquad 1)$$

and the L_{ii} can be unit matrices of dimension m, r and $n-m-r$. The inverse design key matrix has a similar lower triangular form. We consider the construction of standard design key matrices where L_{11} and L_{33} are unit matrices and L_{22} is a lower triangular matrix L with unit diagonal. Write the inverse design key matrix as

$$\begin{vmatrix} I & 0 & 0 \\ B_{21} & L^{-1} & 0 \\ B_{31} & B_{32} & I \end{vmatrix} \qquad 2)$$

The problem reduces to that of finding matrices S of the form

$$S = \begin{vmatrix} B_{21} & L^{-1} \\ B_{31} & B_{32} \end{vmatrix} = (C_1\ C_2) \qquad 3)$$

which are suitable for constructing useful fractional designs with confounding. We construct two dimensional basic arrays from which matrix S is extracted; the arrays are suitable for several designs.

Example 3.6 Consider the basic array T :

1	1	1	0	0
1	0	1	1	0
0	1	1	1	1
1	1	0	1	1
1	0	0	1	0
0	1	0	1	1

A 2^6 factorial experiment is required in four blocks of four units. Let S be formed by the first four rows and five columns of array T. The columns of S correspond to plot factors I_1, I_2, P_1, P_2 and P_3, the first two being dummy factors identified with the fraction. The rows of S correspond to (basic) treatment factors C, D, E, F; generators for the fraction and confounded effects are therefore ACDF, BCEF and CDE, DEF respectively. Note that because of randomization the form of the last column is irrelevant and the same array can be used for blocks of two or eight units.

The construction of basic arrays is performed in two parts. Firstly the columns corresponding to the fraction (matrix C_1) are formed without regard to confounding, then the columns corresponding to the confounded effects (matrix C_2) are chosen. The choice of matrix C_1 is closely related to the constructions of Das (1964) and Box and Hunter (1961).

Example 3.7 For 2^n factorial experiments in blocks of 16 units ($n-m=4$) there exists a 4×11 array T such that if the first $n-4 \geq 0$ columns of T are chosen to form C_1, then

i) for $n \leq 8$ the design has resolution IV;

ii) for $8 < n \leq 15$ the design has resolution III.

(The array is given in Appendix A3.4.) It is not possible to further arrange the columns of the array so that for $n = 5$ the design has resolution V. The array T is thus suitable for generating systems of defining contrasts for $6 \leq n \leq 15$ factors. Over this range the system of defining contrasts is in some sense as 'good' as any other.

A definition of a 'best' system of defining contrasts is given in Appendix A3.2. (The definition is such that a best system is not unique; any permutation of the treatment factors will yield another, equivalent, best system. Furthermore, two best systems are not necessarily equivalent). Under this definition all the defining contrast systems for $n-m=4$ and $6 \leq n \leq 15$ mentioned in Example 3.7 are best but the four factor defining contrast for $n = 5$ factors is not. Commonly, as in this example, there exists no array T which is suitable for both $m = 1, 2$ and $m \geq 4$; similar difficulties arise with small values of $n-m$. A useful compromise procedure for constructing matrices C_1 is given below.

3.4.1 Choice of matrix C_1 For two-level factorial experiments two approaches to the construction of C_1 are suggested:

A. For fixed $n-m$ construct the basic array column by column using the method of Box and Hunter (1961) or Das (1964);

B. For fixed m construct the basic array row by row.

Method A is most appropriate for small $n-m$ and large n or designs with resolution IV or less and method B for small m and large n or designs with resolution five or more.

The following table, showing for designs up to 256 units the maximum number of factors n allowable to achieve the required resolution, may be used as a rough guide to the best construction; above the dotted line use method A otherwise method B. (An alternative table is given in Appendix A3.4.)

$n-m$	4	5	6	7	8
resolution = III	15	31	63	127	255
IV	8	16	32	64	128
V	5	6	8	11	17
VI	-	6	7	9	12

Method A Methods of constructing defining contrast systems for designs of resolution III or IV noted by Das (1964) (§3.3) may be used. For the resolution V design with $n = 17$, $n-m = 8$ either method A or B may be used; the defining contrast scheme given in Appendix A3.4 is equivalent to that presented by Addelman (1965).

Method B Construction is aided by the results of Appendices A3.2 and A3.3. In particular, all possible distinct rows (other than the row of zeros) are made to occur as equally often as possible. Theorem A3.3.1 is very useful when constructing matrices for $m \leq 5$ when $p = 2$ and $m \leq 3$ when $p = 3$.

3.4.2 Choice of matrix C_2

The confounded effects in the basic arrays given in Appendix A3.4 were determined by inspection. The first confounded effect was chosen as the 'best' available (for all choices of C_1) then the second confounded effect chosen as the best available given the

presence of the first etc. Though this construction is apparently appropriate for a succession of nested block strata it is in practice suitable for a single block stratum. The best choice of confounded effects for method A changes markedly over the range $m = 1-3$, but as method B is to be preferred in this range little is lost. For $p = 3$ method A has been preferred and in Appendix 3.4 different confounded effects are given for $m = 2,3$ and $m = 4,5$.

3.5 Selecting defining contrasts

When the block size is small compared to the total number of treatments and some factors are more likely to interact than others, use of a 'best' system of defining contrasts is not appropriate and designs given in standard tables, such as Cochran and Cox (1957) may not be suitable. In this section we present algorithms for selecting defining contrasts in general fractional or confounded p^n factorial experiments.

In §3.5.1 some definitions are given, and in §3.5.2 the special case $p = 2$ is outlined with special reference to the paper by Franklin and Bailey (1977) given in Appendix A3.5. The basic algorithm is extended to general prime values in Appendix A3.6 and in various other ways in Appendices A3.6.1 to A3.6.3. Some examples are given in §3.5.3.

Note: Throughout the discussion we use a dummy block category called the fraction. A fractional design may thus be considered as having the same block structure as the confounded design with the same defining contrasts. The problems of finding confounded effects and of finding defining contrasts can be considered as a single problem, a simplification which becomes important with more complex block structures.

3.5.1 Definitions

The $(p^n - 1)/(p - 1)$ treatment effects are divided into sets:

i) the effects P_i for which estimates are required (called the requirements set by Greenfield (1976));

ii) the other unknown non-negligible effects Q_i ;

iii) the known or negligible effects R_i .

For a single replicate block design the effects ineligible as defining contrasts of the confounding scheme are simply the P_i . For a fractional design the effects ineligible as defining contrasts are the P_i and the generalized interactions $P_i P_j^r$ and $P_i Q_j^r$, $r = 1, 2, \ldots, p-1$. Those effects ineligible for choice as confounded effects/defining contrasts are called the ineligible effects: the remaining effects are called eligible. For convenience the mean effect I is deemed both eligible and ineligible.

To avoid misclassification when effects are checked for eligibility, we adopt the convention that if an effect E is ineligible then all $p-1$ ways of writing that set of $p-1$ degrees of freedom, i.e. $E, E^2, \ldots, E^{p-1}$ are ineligible. Once the set of ineligible effects have been specified there is no difference between the case of a single replicate design with confounding and that of a fractional replicate.

The algorithm described in Appendix A3.6 constructs a set of $(p^m-1)/(p-1)$ eligible defining contrasts, if one exists, by searching for a set of m different eligible effects, none of which can be expressed as an interaction among the remaining $m-1$ effects (i.e. a set of m defining contrast generators), such that all interactions among the m effects are eligible. The set consisting of the mean effect I and all $p-1$ ways of writing each of the $(p^m-1)/(p-1)$ defining contrasts forms a group of order p^m ; such a group is called

a group of (eligible) defining contrasts. The purpose of the algorithm can be regarded as finding all groups of eligible defining contrasts of order p^m.

The algorithm is closely related to the method of Das (1964) outlined in §3.3 and we use Das's terminology of basic and added factors. Any effect involving only basic factors is called a basic effect.

3.5.2 Two-level experiments

Greenfield (1976) presents an algorithm for selecting defining contrasts in two-level experiments. His algorithm is intended for use with fractional but not confounded designs. He defines the requirements set as in §3.5.1 but does not define ineligible effects; all main effects, first-order interactions and interactions between pairs of treatments in the requirements set are effectively ineligible. The definition of ineligible and eligible effects by Franklin and Bailey (1977, see Appendix 3.5) gives greater versatility and enables fractional designs and confounded designs to be treated in the same way. Moreover, by defining for each effect the strata in which it is ineligible for selection as a defining contrast the procedure can be extended to the important class of fractional designs with confounding and other multiple strata designs.

Franklin and Bailey (1977, p. 325) point out a flaw in Greenfield's algorithm which can result in a failure to find the smallest possible fraction. The algorithm selects defining contrast generators of the form B_iA_j where B_i is a basic effect and A_j is an added factor. All defining contrasts must contain an added factor i.e. no basic effect can be a defining contrast. Greenfield recommends that the first m factors be taken as the basic factors but this procedure

fails if the smallest design contains a defining contrast involving these factors only. Franklin and Bailey (1977, p. 323) suggest

a) selecting a set of basic factors for which no effect is eligible,

or b) searching through all possible sets of basic factors.

An improved procedure is suggested in Appendix A3.6.3.

3.5.3 The algorithm and its extensions

The algorithm presented in Appendix A3.6 is a direct extension to p^n designs of that described in §3.5.2 for 2^n designs. An explanation of the algorithm is also given. The algorithm is in its most general form; in Appendix 3.6.1 some modifications are given which improve the algorithm for commonly occurring problems. In Appendix A3.6.2 the extension of the use of the algorithm to designs with a crossed block structure is illustrated and a procedure for applying the algorithm to any simple block structure is outlined. Selection of a suitable set of $n-m$ basic factors can be tedious for large n and $n-m$ and in Appendix A3.6.3 a modification to the procedure for selecting the basic factors is given.

Example 3.8 Most important features of the algorithm in Appendix A3.6 and its extensions are demonstrated by this example. A single design is required for a 3^5 factorial experiment to estimate main effects and the interactions AB, BC, all other effects being assumed negligible. The design is to be a fractional replicate of smallest possible size, with the plots arranged in blocks of smallest possible size.

Stage 1:-Fraction The effects ineligible as defining contrasts are all main effects and two factor interactions, and all components of ABC, ABD, ABE, BCD and BCE . The group generated by A, B, C is all

ineligible so we take $m = 5 - 3 = 2$ and the single set of basic factors A, B, C . As described in Appendix A3.6.1, the table constructed in Step 4 can be abbreviated to:

Basic Effects	Added Factors D	E
AB	-	-
AB^2	-	-
AC	ACD	ACE
AC^2	AC^2D	AC^2E
BC	-	-
BC^2	-	-
ABC	ABCD	ABCE
ABC^2	ABC^2D	ABC^2E
AB^2C	AB^2CD	AB^2CE
AB^2C^2	AB^2C^2D	AB^2C^2E

ACD may be chosen from the first column giving $G_1 = I, ACD, A^2C^2D^2$. The rest of row AC can be declared ineligible, and AC^2E chosen from the next column. Now $(AC^2E)(ACD) = A^2DE$ and $(AC^2E)(A^2C^2D^2) = CD^2E$ are both eligible, so there is a 1/9-replicate available with defining contrasts ACD, AC^2E, A^2DE, CD^2E .

Stage 2:-Blocks The effects ineligible for being confounded with blocks are the main effects and all components of AB, BC . The previous added factors D, E are removed and replaced by their aliases A^2C^2 and A^2C . The list of ineligible effects consists of A, B, C, A^2C^2, A^2C, AB, AB^2, BC, BC^2 and their powers.

The group generated by A, B is all ineligible, so we take $m = 3 - 2 = 1$ and the single set of basic factors A, B . The table can again be abbreviated.

Basic Effects	Added Factor C
AB	ABC
A^2B^2	A^2B^2C
AB^2	AB^2C
A^2B	A^2BC

ABC can be confounded with blocks, giving a fraction in three blocks of nine units, with defining contrasts ACD, AC^2E, A^2DE, CD^2E and confounded effects ABC, A^2BC^2D, BD^2, A^2BE, BC^2E^2, BCDE, $A^2BCD^2E^2$, ABC^2D^2E, $ABDE^2$.

Example 3.9 This example demonstrates the procedure used when no group of basic effects is completely ineligible. All designs are required for a 2^5 factorial experiment in four blocks of eight units in which no main effect or two factor interaction is confounded. Thus $m = 2$ is specified. Since all three factor interactions are eligible there is no ineligible basic effects group of size 2^3 . We take A, B, C as basic factors. The table may be abbreviated to:

Basic Effects	Added Factors D	E
AB	ABD	ABE
AC	ACD	ACE
BC	BCD	BCE
ABC	ABCD	ABCE

Twelve possible groups of confounded effects are obtained by choosing elements from different rows of the two columns.

The basic effect ABC is eligible, and none of the groups of confounded effects found so far contains ABC . Factor A is removed and replaced by A(ABC) = BC in the ineligible effects. Thus only, BDE, CDE, BCDE remain eligible, and any of these may be chosen as the single confounded effect at the second stage. There are therefore three more eligible groups of confounded effects: ABC, BDE, ACDE; ABC, CDE, ABDE; ABC, BCDE, ADE .

Appendix A3.1

Equivalent key matrices for p^n designs (p prime)

The formation of libraries for key matrices is facilitated by the adoption of standard matrix forms. Let designs D_1 and D_2 have the same (simple) block and treatment structure, but be formed from different design keys. The design keys are said to be equivalent if the allocation of treatment effects to block strata is the same for both i.e. the designs have the same defining contrast system. In this appendix we show that two key matrices K_1 and K_2 are equivalent if $K_2 = U K_1$ for suitable non-singular upper triangular matrices U.

Example For a 3^3 design in 3 blocks of 9 units the design keys $P_1 = ABC$, $P_2 = B$, $P_3 = C$ and $P_1 = A^2B^2C^2$, $P_2 = B$, $P_3 = C$ are equivalent and in the associated designs both ABC and $A^2B^2C^2$ are confounded with blocks.

Let design D_1 contain b block factors with p^{m_i} levels ($i = 1 \ldots b$) then the $n = \Sigma m_i$ rows of the key matrix can be partitioned into b sets, the ith set containing m_i rows. Choose block factor B_i ; we consider two cases

i) B_i is not nested in any other factor.

ii) B_i is nested in one or more factors.

Case i): Let the $s = m_i$ plot effects $P_1P_2 \ldots P_s$ corresponding to block factor B_i be identified with treatment effects $E_1, E_2 \ldots E_s$. Any set of s linearly independent vectors from the space generated by the E_k ($k = 1 \ldots s$) can be identified with the P_k without changing the equivalence class. Each set forms a basis for the vector space and changing bases is equivalent to multiplication by a non-singular matrix. The columns of the inverse design key matrix K_1^{-1} corresponding to block factor B may be right multiplied by a non-

singular block diagonal matrix with partitions determined by the m_i then K_1 and K_2 are equivalent design key matrices.

Case ii): Let block factor B_i be nested in B_j (where B_j can denote the product of factors). The s plot effects P_k ($k = 1...s$) can be identified with any choice of s linearly independent treatment vectors from the $m_j m_i - m_j$ confounded in stratum $B_j.B_i$ leaving the defining contrast system unchanged. Now if treatment effects E_1 and E_2 are confounded in strata B_j and $B_j.B_i$ respectively then $E_2 E_1^r$ ($r \neq 0$) is confounded in stratum $B_j.B_i$ and may be identified with any of the P_k. Therefore, if D_j and D_i are submatrices of the inverse key matrix formed by the columns corresponding to block factors B_j and B_i, then replacement of D_i by the matrix $D_j C_j + D_i C_i$ where C_i is a non-singular matrix leaves the defining contrast system unchanged. This operation is equivalent to right-multiplying the inverse key matrix by a block upper diagonal matrix, with unit diagonal apart from C_i and with C_j above the diagonal determined by the locations of block factors B_j and B_i.

The arguments are readily applied to any simple block structure and therefore a defining contrast system is unchanged by left multiplication of the key matrix by the non-singular block upper triangular matrix.

$$C = (\delta_{ij}\, C_{ij})$$

where δ_{ij} is element (j, i) of the lower triangular nesting matrix M (§1.3) and C_{ij} is a block submatrix of C defined by the locations of block factors i and j.

Appendix A3.2

Choosing 'best' defining contrasts systems

In this appendix we describe a method for constructing standard defining contrast systems when all factors have equal importance.

Definition: For a p^n design confounded in p^m blocks of p^{n-m} units the defining contrast (confounding) system is best if

i) The total of the lengths of the defining contrasts (i.e. word lengths) is the maximum achievable;

ii) No design satisfying i) has a smaller variance among the word lengths;

iii) No design satisfying i) and ii) has a higher resolution.

Constructing best systems

Generators for any suitable defining contrast system can be arranged as columns of an $n \times m$ matrix M with elements residues modulo p. We demonstrate that for best systems all distinct rows, except that containing only zeros (the null row), should occur as equally often as possible. (Two rows are called distinct if one is not a multiple of the other.)

Step 1: If a factor occurs in any defining contrast it occurs in exactly $(p-1)\,p^{m-1}$ defining contrasts. The condition i) is thus satisfied by all systems in which all factors appear i.e. M does not contain a null row. The total length is

$$\Sigma d_i = n(p-1)\,p^{m-1} .$$

Step 2: Let e_j be the number of occurrences of all multiples of the jth possible distinct row of M (excluding the null row). We show

that the variance of the word lengths is proportional to the variance of the e_j . Let the columns of M (i.e. the defining contrast generators) be denoted g_1 , g_2 , ... g_m . Each defining contrast including the mean can be expressed as $\Sigma a_k g_k$ $(0 \leq a_k \leq p-1)$ and can be ordered lexicographically by the a_k . (This is the order in which they are normally generated.) The p^m possible rows of M may also be listed in lexicographical order. Let the row in M corresponding to factor U be the jth in the lexicographical list then the power of factor U in the ith defining contrast is given by the (i,j)th element of matrix A_m where

$$A_1 = \text{multiplication table of residues modulo } p$$

$$A_m = A_{m-1} \otimes J + J \otimes A_{m-1} \qquad m > 1 \qquad 2)$$

where J is the $p^{m-1} \times p^{m-1}$ matrix with all elements unity.

Example Let CDE^2F^2 and CD^2F^2 be confounded with blocks then

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 0 \\ 2 & 2 \end{pmatrix}$$

and

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 2 & 1 & 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 0 & 1 & 1 & 2 & 0 \\ 0 & 2 & 1 & 2 & 1 & 0 & 1 & 0 & 2 \end{pmatrix}$$

Factor E corresponds to row (2 0) in M and hence the seventh column of A_2 . Its power in the sixth defining contrast = $(CDE^2F^2)(CD^2F^2)^2$ is 2 .

Adopting the ordering (01, 10, 11, 12), matrix B and vectors $\underline{d}$, $\underline{e}$ (see below) are

$$B = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \qquad \underline{e} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix} \qquad \text{and} \qquad \underline{d} = \begin{pmatrix} 3 \\ 4 \\ 3 \\ 2 \end{pmatrix}$$

The word length of the ith defining contrast is obtained by summing the number of factors associated with non-zero elements of the ith row of A_m. In practice this is easily done by substituting 1 for any non-zero element in A_m, forming a $p^m \times 1$ vector of counts $\underline{c}$ giving the number of factors associated with each possible row of M then the vector of defining contrast lengths $\underline{b}$ is given by

$$\underline{b} = A\underline{c} \qquad 3)$$

where A is the O-1 matrix derived from A_m. Now, the p-1 rows of A corresponding to multiples of the same defining contrast are identical; the p-1 columns which are multiples of one treatment are also. Let P be a $(p^m-1)/(p-1) \times p^m$ orthonormal matrix whose jth row contains $(p-1)^{-\frac{1}{2}}$ in each column corresponding to a multiple of the jth possible distinct non-null row of M then

$$\underline{d} = (p-1)^{-\frac{1}{2}} P\underline{b} = (p-1)^{-\frac{1}{2}} PAP^T P\underline{c} = B\underline{e} \qquad 4)$$

where $B = PAP^T$ is a O-1 matrix, e_j is the count of all p-1 multiples of the jth possible distinct row of M and $\underline{d}_i$ is the word length of the ith distinct defining contrast. (See example above.)

Matrix B has a special form:

i) it is symmetric of order $(p^m-1)/(p-1)$

ii) each row contains p^{m-1} non-zero elements

iii) each pair of rows have $p^{m-2}(p-1)$ elements in common.

B is therefore the incidence matrix of the symmetric BIB design with

$$b = v = (p^m-1)/(p-1) \quad r = k = p^{m-1} \quad \lambda = p^{m-2}(p-1) \; . \qquad 5)$$

It follows that

$$\begin{aligned} \Sigma d_i^2 = d^T d &= e^T B^T B e \\ &= e^T (p^{m-2}(I + (p-1)J) e \\ &= p^{m-2}(\overset{b}{\Sigma} e_j^2 + (p-1)(\overset{b}{\Sigma} e_j)^2) \end{aligned} \qquad 6)$$

Now, $\Sigma d_i = np^{m-1}$ and $\Sigma e_j = n$ so we obtain

$$SS(d_i) = p^{m-2} SS(e_j) \qquad 7)$$

where SS indicates sums of squares corrected for the mean and hence

$$var(d_i) = p^{m-2} var(e_j) \qquad 8)$$

Thus we have shown that the variance of the word lengths d_i is proportional to the variance of the e_j, the number of times the possible distinct rows of M occur. The rows should therefore occur as equally often as possible.

Step 3: We conjecture that there is a maximum resolution defining contrast system (i.e. the one with $\max(\min(d_i))$ which is also a system with minimum variance. (In justification it is noted that $\bar{d}$ is constant so a maximum resolution system is one which minimizes $(\bar{d} - d_{min})$. Let S_1 be the set of minimum variance systems and S_2 the set of minimum half-range systems then if for some n and m the conjecture does not hold the two sets have no members in common.)

A minimum variance design need not have maximum resolution.

Example Let the defining contrast system for a 2^8 design in blocks of 16 plots be represented by the matrix M where

$$M^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Excluding the null row there are 15 possible rows of M, eight of which occur. The vector $\underline{c}$ in 3) above contains 0's and 1's and the defining contrast system therefore has minimum variance. This design has resolution 3 whereas the design formed by replacing element $M^T_{1,8}$ by zero also has minimum variance but resolution 4.

Appendix A3.3

Constructing best defining contrast systems

We present as a theorem two constructions for best defining contrast systems; both theorems are used in forming the arrays of Appendix A3.4.

Theorem A3.3.1 The following designs can be constructed for all values of m :

i) for $n = (p^m-1)/(p-1)$, a p^{n-m} design with all defining contrasts of length p^{m-1} ;

ii) for $n = p^{m-1}$ a p^{n-m} design with one defining contrast of length p^{m-1} and all others of length $(p-1)p^{m-2}$. No other (non-isomorphic) set of defining contrasts for the same n and m has resolution $(p-1)p^{m-2}$.

Proof by construction. (See Appendix A3.2 for notation.)

i) There are at most $(p^m-1)/(p-1)$ distinct rows of the matrix M (Appendix A3.2) such that none can be expressed as a multiple of another. Form M from these rows then $e_j \doteq 1$ for all j and $\text{var}(d_i) = 0$ so $d_i = \bar{d} = p^{m-1}$ for all i . Clearly no non-isomorphic system can have resolution $\bar{d} = p^{m-1}$.

ii) Select any defining contrast from the set constructed in part i) . Delete all factors and rows of the matrix M for which this defining contrast has value zero. From the BIB property noted at 5) in Appendix A3.2 the selected contrast has length p^{m-1} and all others have length $(p-1)p^{m-2}$.

Call this system of defining contrasts S . Consider any system of defining contrasts for $n = p^{m-1}$. Let the word length of the defining contrasts be k_i and form $m_i = k_i - (p-1)p^{m-2}$. Now, if $m_i \geq 0$ for all i then because the mean is fixed the

maximum variance is achieved only by system S (or systems isomorphic to it). But by construction S is the minimum variance system so all other systems have variance as large or larger than S. This implies that other systems must contain at least one $m_i < 0$. It follows that S and its isomorphisms are the only systems for $n = p^{m-1}$ of resolution $(p-1)p^{m-2}$.

Appendix A3.4

Standard matrices for fractional designs with confounding

The arrays presented here yield matrices of the form

$$(C_1 C_2) \ .$$

When modified to the form

$$\begin{pmatrix} I & 0 \\ C_1 & C_2 \end{pmatrix}$$

the first m columns are the generators of the defining contrasts of the fraction, the last r columns the generators of the confounded effects (§3.4). Two sets of arrays are presented A and B . For construction A the number of rows $n-m$ is fixed; m , and thus n , is fixed by selecting m columns moving progressively leftward from the partition. For construction B the number of columns m is fixed; $n-m$, and thus n , is determined by selecting the first $n-m$ rows of the matrix. The matrices C_2 determine the order in which the treatments are allocated to units and the effect of C_2 on confounding is determined by the randomization applied.

Construction A

$n-m = 4$

	1	0	1	1	1	0	0	0	1	1	1	1	0
	1	1	0	0	1	1	0	1	0	1	1	1	1
	1	0	1	0	0	1	1	1	1	0	1	1	0
	1	1	0	1	0	0	1	1	1	1	0	1	1
resolution	3	3	3	3	3	3	3	4	4	4	4		

$n-m = 5$

	1	0	0	1	1	1	0	1	0	1	1	1	0
	1	1	0	0	1	1	1	0	1	0	1	1	1
	1	1	1	0	0	0	1	1	0	1	1	1	1
	0	1	1	1	0	1	0	1	1	0	1	0	1
	0	0	1	1	1	0	1	0	1	1	1	0	0
resolution	4	4	4	4	4	4	4	4	4	4	5		

n-m = 6

	0	1	0	1	1	1	1	1	1	0	1	0	0
	1	1	0	0	1	1	1	1	0	1	1	1	0
	1	0	0	1	1	1	1	0	1	1	1	0	1
	0	0	1	1	1	1	0	1	1	1	0	1	1
	1	0	1	0	1	0	1	1	1	1	0	0	1
	0	1	1	0	0	1	1	1	1	1	0	1	0
resolution	4	4	4	4	4	4	4	4	4	6			

n-m = 7

	1	0	1	1	1	1	0	1	1	0	0
	1	1	0	1	1	1	1	0	1	1	0
	1	0	1	0	1	1	1	1	0	1	1
	1	1	0	1	0	1	1	1	1	0	1
	1	1	1	0	1	0	1	1	0	0	0
	1	1	1	1	0	1	0	1	0	0	1
	1	1	1	1	1	0	1	0	0	1	0
resolution	4	4	4	4	4	4	6	6			

n-m = 8

	1	1	1	0	1	0	0	0	1	1	0	0
	1	1	0	1	0	0	0	1	1	0	1	0
	1	0	1	0	0	0	1	1	1	0	1	1
	1	1	0	0	0	1	1	1	0	1	0	0
	1	0	0	0	1	1	1	0	1	0	0	1
	1	0	0	1	1	1	0	1	0	1	1	0
	1	0	1	1	1	0	1	0	0	0	1	1
	1	1	1	1	0	1	0	0	0	1	0	1
resolution	5	5	5	5	5	5	5	5	5			

Note: Array chosen to achieve resolution V . For $m \leq 4$ use method B . For $m \geq 10$ use a resolution IV array based on columns with an odd number of units.

n-m = 9

To the matrix for n-m = 8 append the row

	1	1	1	1	1	1	1	1	1	0	0	0

Construction B

m = 1

						resolution
1	1	0	0	0	0	2
1	0	1	0	0	0	3
1	1	1	1	0	0	4
1	0	0	1	1	0	5
1	1	0	1	1	1	6
1	0	1	1	0	0	7
1	1	1	0	0	1	8
1	0	0	0	1	0	9
1	0	1	0	1	1	10
1	1	0	0	1	1	11
1	1	1	1	1	1	12

m = 2

						resolution
1	1	1	0	0	0	2
1	0	1	1	0	0	2
0	1	1	1	1	0	3
1	1	0	1	1	1	4
1	0	0	1	0	1	4
0	1	0	1	1	0	5
1	1	1	1	0	1	6
1	0	1	1	0	1	6
0	1	1	1	0	1	7
1	1	0	0	0	0	8

m = 3

						resolution
1	1	1	1	0	0	2
1	1	0	0	1	0	2
1	0	1	0	0	1	3
0	1	1	1	1	1	4
1	0	0	1	0	1	4
0	1	0	1	0	0	4
0	0	1	1	1	0	5
1	1	1	0	1	1	6
1	1	0	1	1	0	6
1	0	1	1	1	1	7
0	1	1	0	0	0	8

m = 4

						resolution
0	1	1	1	1	0	1
1	0	1	1	0	1	2
1	1	0	1	0	0	3
1	1	1	0	1	1	4
1	0	0	1	1	0	4
0	1	1	0	0	1	4
1	0	1	0	0	0	5
0	1	0	1	1	1	6
1	1	0	0	1	0	6
0	0	1	1	0	0	7
1	1	1	1	0	1	8

m = 5

						resolution
1	1	1	1	1	1	2
1	1	1	0	0	0	2
1	0	0	1	1	0	2
1	1	0	0	1	1	3
1	0	1	1	0	1	4
0	0	1	1	1	0	4
0	1	0	1	1	1	4
0	1	1	0	1	0	5
0	1	1	1	0	1	6
1	1	0	1	0	0	7
1	0	1	0	1	1	8

m = 6

							resolution
1	1	1	1	1	1	1	2
1	0	1	0	1	0	0	2
1	0	0	1	0	1	0	2
0	1	1	0	0	1	1	3
0	1	0	1	1	0	1	4
1	1	0	1	0	0	0	4
0	0	1	1	1	0	0	4
1	0	1	0	0	1	1	5
0	1	0	0	1	1	0	6

Choice of construction A or B

The following diagram indicates the preferred method of construction for values of n and n-m (X = either method).

n-m \ m	<4	4	5	6	>6
≤4	A	A	A	A	A
4	B	X	A	A	A
5	B	X	X	X	A
6	B	X	X	X	A
7	B	X	X	X	A
8	B	B	X	X	A
9	B	B	B	B	A

Matrices for p = 3

m = 1

1	1	0	0
1	2	1	0
1	0	1	1
1	1	1	0
1	2	2	1
1	0	2	1

m = 2,3

2	1	1	1	0	0
0	2	1	1	1	0
1	0	2	1	1	1
1	1	0	1	2	1
1	1	1	2	1	0
1	2	1	0	2	1

m = 4,5

1	0	2	1	1	1	0	0
1	1	0	2	1	2	1	0
1	1	1	0	2	1	1	1
1	2	1	1	0	2	1	2
2	1	1	1	1	0	1	0
0	2	1	1	1	2	0	0

Selection of Defining Contrasts and Confounded Effects in Two-level Experiments

BY

M. F. FRANKLIN and R. A. BAILEY

Reprinted from

THE JOURNAL OF THE ROYAL STATISTICAL SOCIETY

SERIES C (APPLIED STATISTICS)

Volume 26, No. 3, 1977

(pp. 321–326)

PRINTED FOR PRIVATE CIRCULATION

1977

Appl. Statist. (1977),
26, *No.* 3, pp. 321–326.

Selection of Defining Contrasts and Confounded Effects in Two-level Experiments

By M. F. FRANKLIN and R. A. BAILEY†

ARC Unit of Statistics, Edinburgh, Scotland *University of Edinburgh, Scotland*

[Received March 1977. Revised May 1977]

SUMMARY

In factorial experiments the selection of defining contrasts and effects to be confounded with blocks is not always straightforward and plans presented in standard texts are not necessarily suitable. A systematic procedure for selecting suitable defining contrasts has been presented by Greenfield (1976). In this paper we show that Greenfield's algorithm does not always generate the smallest possible fraction, and give a more general procedure which is suitable for selecting defining contrasts and confounded effects.

Keywords: ADDED FACTOR; ALGORITHM; ALIAS; BASIC FACTOR; CONFOUNDING; DEFINING CONTRASTS; DESIGN KEY; FACTORIAL DESIGN; FRACTIONAL REPLICATION; 2^n EXPERIMENTS

1. INTRODUCTION

WHEN the block size is small compared to the total number of treatments in a factorial design the selection of defining contrasts and effects to be confounded with blocks is not straightforward and designs given in standard texts, such as Cochran and Cox (1957), are not always suitable. In this paper we present a standard procedure for generating suitable single replicate or fractional replicate 2^n factorial designs once the requirements have been expressed.

Greenfield (1976) gave a simple algorithm for finding a suitable set of defining contrasts for a fractional replicate of a 2^n factorial design from which certain specified main effects and interactions needed to be estimated, all other effects being negligible. His algorithm cannot be used for designs which involve confounding. In Section 2 we present an algorithm for finding all designs for n factors at two levels, either 2^{-m} fractions or single replicates in blocks of 2^{n-m}, where certain specified main effects and interactions are required not to be defining contrasts or confounded effects. In Section 4 we show how a repeated application of the algorithm can be used for fractional replicates with confounding. For simplicity, where no confusion is likely we shall refer to confounded effects as defining contrasts in Sections 2 and 3, as the procedure is similar for both.

Greenfield (1976) said that "it would not be difficult to express [his solution] in more rigorous mathematical terms". However, in the justification (Section 3) for our algorithm we show that Greenfield's algorithm will not always generate the smallest possible fraction.

2. THE ALGORITHM

In this section we adopt Greenfield's style of presentation and the same example, in order to facilitate comparisons between the papers. The example is a fractional replicate of a 2^5 design, and for this example his objective is to find "the smallest possible balanced fractional design that can be used to estimate each of the main effects [A, B, C, D and E] and also the effects of the interactions AB and BE, assuming all other interactions are negligible". The

† Science Research Council Research Fellow on secondment from the Open University.

objective of the algorithm presented below is to find all designs with the smallest possible block size such that no member of a given set of effects is selected as a defining contrast (or is confounded with blocks).

In this algorithm we input or generate the set of effects which are not eligible for choice as defining contrasts. For convenience we call these *ineligible effects*. The effects eligible for choice as defining contrasts, we call *eligible effects*. (Later discussion will be simplified if we assume the mean effect I is a member of both sets.) Greenfield calls the set of effects which he wishes to estimate the *requirements set*. For a given requirements set all effects in the set are ineligible for selection as confounded effects or defining contrasts and all generalized interactions between pairs of effects in the set are ineligible for selection as defining contrasts. Thus for our example the requirements set is

$$\{A, B, C, D, E, AB, BE\}$$

and the ineligible set is

$$\{I, A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, E, AE, BE, ABE, CE, BCE, DE, BDE\}.$$

Excluding the mean effect, the 2^n-1 effects may be divided into three sets:

(i) a set of effects A_i for which estimates are required;
(ii) a set of effects B_i which are not negligible but for which no estimates are required;
(iii) a set of effects C_i which are considered negligible.

For a single replicate confounded design, the requirements set and the set of ineligible effects are identical with the set of effects A_i. For a fractional design the set of ineligible effects is the union of the set of effects A_i, the set of all generalized interactions A_iA_j and the set of generalized interactions A_iB_j. However, it is clear from the relationship between the requirements set and the set of ineligible effects described in the last paragraph that if we start from just a requirements set the generalized interactions A_iB_j will be wrongly classified as eligible effects. For the rest of this paper, therefore, we shall work with eligible and ineligible effects. The use of eligible and ineligible effects has one other advantage over the use of a requirements set; once the set of eligible effects has been chosen it is no longer necessary to distinguish between fractional designs and confounded designs.

It is well known that for a set of k factors the effects of these factors together with the mean effect and all interactions among the factors form a group of size 2^k. It is helpful in describing the algorithm for generating a 2^n experiment with blocks of size 2^{n-m} to use the notation of Das (1964) and refer to $n-m$ of the factors as *basic factors* and the remaining m factors as *added factors*. The group of size 2^{n-m} containing all main effects and interactions among the $n-m$ basic factors we shall call the *basic effects group*.

For a 2^{-m} fraction there must be 2^m-1 defining contrasts, which together with the mean effect I form a group of size 2^m. Clearly if we can find a suitable group of 2^m defining contrasts then we can derive a 2^{-m} fraction. The smallest permissible fraction is thus determined by the size of the largest group in the eligible effects set. The central part of the algorithm defines a search procedure for such a group and if there are several groups having this largest size the algorithm will find all of them.

Although the search may start with a block size of one (i.e. initially $m=n$) this will normally cause some wasted effort. If the largest groups in the eligible and ineligible sets have sizes 2^r and 2^s respectively then $r+s \leqslant n$. (The complete effects group, of size 2^n, contains the direct product of these groups, which has size 2^{r+s}.) The amount of searching may therefore be reduced by using the starting value suggested in step 2. In practice it is not difficult to select a suitable starting value for m and an overestimate will not affect the eventual outcome.

The algorithm may be expressed as follows.

Step 1. Input or generate the ineligible effects set.

Step 2. Choose m such that the largest group in the ineligible effects set contains not more than 2^{n-m} members.

Example. The largest groups in the ineligible effects set (e.g. that formed by factors A, B and C and all their interactions and the mean effect) contain eight members, which suggests an initial value of 2 for m.

Step 3. Select a (new) set of $n-m$ basic factors and form the basic effects group.

Step 4. Form a two-way table with 2^{n-m} rows headed by the basic effects and m columns headed by the added factors. In the ith row and jth column of the table note whether the interaction, $B_i A_j$, between the ith basic effect B_i and the jth added factor A_j is an eligible effect.

Example (− denotes an ineligible effect):

		Added factors	
		D	*E*
	I	−	−
	A	−	−
	B	−	−
Basic effects	*AB*	−	−
	C	−	−
	AC	*ACD*	*ACE*
	BC	*BCD*	−
	ABC	*ABCD*	*ABCE*

Steps 5 to 10 of this algorithm describe a search for m defining contrasts, one selected from each of the m columns of this table, which form a basis for the largest group of defining contrasts in the eligible effects set. It is convenient to have a pointer for each column which indicates which is the next available effect in that column, and also a number indicating the column currently being searched, where the columns are numbered 1 to m.

Step 5. Define a starting position for a search of the table, i.e. column number 0 and a defining contrasts group containing only the mean effect I.

Step 6. Move to the start of the next column (i.e. increase the column number by one) and initialize the pointer for this column.

Step 7. Select the next available effect in the current column and adjust the pointer appropriately. If the elements in the current column have been exhausted move to step 10.

Step 8. Check whether all generalized interactions between the effect selected and the defining contrasts group are eligible effects. If not, return to step 7.

Step 9. Extend the defining contrasts group by the selected effect and the interactions. If the last column has been reached an acceptable set of defining contrasts has been found so output this set (or the full group if desired) and return to step 7, otherwise return to step 6.

Step 10. If the current column is the first move to step 11; otherwise move to the previous column (but do not reinitialize the pointer) and return to step 7.

Step 11. The search procedure using the current basic factors set is complete; if a new set of basic factors is available return to step 3.

Step 12. If a suitable design has been found or if $m = 1$ terminate the search, otherwise decrease m by one (to increase the block size) and return to step 3.

Example. The algorithm causes the table to be searched in the following way. The first eligible effect in the first column, ACD, is selected. The second column is then searched for any eligible effect whose generalized interaction with ACD is also eligible. There are none, so ACD is replaced by the next eligible effect in the first column, BCD. The search of the second column is repeated, this time for any eligible effect whose generalized interaction with BCD is also eligible. There are two such effects, ACE and $ABCE$. As there are no more columns to search $I = BCD = ACE = ABDE$ and $I = BCD = ABCE = ADE$ are suitable groups of defining contrasts. When the search of the second column has been completed BCD is replaced

by the next eligible effect in the first column, $ABCD$, and the search of the second column repeated. No suitable effects are found and as the first column has been exhausted the search procedure is terminated. (It will be shown later that there is no need to choose another set of basic factors in this case.)

3. JUSTIFICATION

The algorithm described in Section 2 produces all suitable designs with blocks of smallest possible size and no others. Firstly, it is clear that whenever all the columns have been exhausted an acceptable set of defining contrasts has been found. For each added factor A_j we have selected a defining contrast containing just A_j among the added factors: thus we have selected m defining contrasts which are independent in the sense that no one of them can be expressed as a product of the others. Moreover, we have checked in step 8 that all generalized interactions among these defining contrasts are eligible effects, so that the group formed by these defining contrasts, their generalized interactions and the mean effect I is suitable and has size 2^m.

Secondly, every suitable design can be generated by this method. Suppose the design has m independent defining contrasts $D_1, \ldots, D_m$. If we can express these as $D_i = B_i A_i$, where $A_1, \ldots, A_m$ are m distinct factors (the added factors) and $B_1, \ldots, B_m$ are effects involving only the other factors (the basic factors), then it is clear that this design will be produced by the algorithm when the basic factors are taken to be all those other than $A_1, \ldots, A_m$.

If we input $D_1, \ldots, D_m$ into the following algorithm it finds a set of added factors A_i and produces an equivalent set of defining contrasts which can be written in the form $D_i = B_i A_i$ as above.

Step 1. Put $i = 0$.

Step 2. Increase i by 1. Choose a factor A_i that appears in D_i. If $i = m$ go to step 4.

Step 3. For $j = i+1, \ldots, m$ replace D_j by $D_i D_j$ if A_i appears in D_j. This ensures that we still have an independent set of defining contrasts but that A_i does not occur in D_j for $j > i$. Return to step 2.

Step 4. For $j = 1, \ldots, i-1$ replace D_j by $D_i D_j$ if A_i appears in D_j. This ensures that A_i does not occur in D_j unless $i = j$.

Step 5. If $i > 2$ decrease i by 1 and return to step 4. If $i = 2$ stop.

At this stage the m factors A_i are the added factors, as each A_i occurs in exactly one of the m defining contrasts D_i, which contains no other factor A_j. Note that steps 1 to 3 of this algorithm are essentially the same as part of an algorithm presented by Berger (1972).

For example, the above algorithm applied to the 2^{5-2} design with defining contrasts ABC, BDE, $ACDE$ might run as follows: defining contrasts $ACDE$, BDE; choose E from $ACDE$; defining contrasts $ACDE$, $BDE \times ACDE = ABC$; choose C from ABC; defining contrasts $ACDE \times ABC = BDE$, ABC; added factors are E, C, basic factors are A, B, D. It is impossible in this case to have D, E or A, C as the added factors.

Note that if the D_i are written as the rows of an $m \times n$ matrix with (i,j) entry equal to one if factor F_j occurs in D_i, 0 otherwise, then the above algorithm describes the reduction of the matrix to the form $(I \quad X)$, where I is the $m \times m$ identity matrix and X is a $m \times (n-m)$ matrix, by elementary row operations followed by a permutation of columns to bring the added factors to the left. Note also that the treatment combinations in the principal block may be derived from the columns of the matrix $\begin{pmatrix} X \\ I \end{pmatrix}$. For a 2^{-m} fractional design the transpose of this matrix may be used as the design key matrix for generating this design. The design key method of generating factorial designs was described for 2^n designs by Patterson (1965) and for asymmetrical factorial designs by Patterson (1976).

In the justification given above it was shown that although there is a suitable set of basic factors for every design it is not always possible to generate all suitable designs from a given set of basic factors. In Greenfield's algorithm the first $n-m$ factors are chosen as the basic factors and the algorithm will thus not always produce the smallest possible fraction. It will fail, for example, with the requirements set $\{A, B, C, D, E, AD, AE\}$. The algorithm should therefore be modified to allow for the selection of alternative basic factor sets.

For a particular selection of basic and added factors it is not possible using the algorithm given in Section 2 to generate any design in which a basic effect is a defining contrast. However, it is possible to generate all designs for which no basic effect is a defining contrast. It follows that if the basic factors can be selected such that no basic effect is an eligible effect then all suitable designs can be generated and it is not necessary to choose any other set of basic factors. In the example used in Section 2 no interaction involving only factors A, B and C is an eligible effect and thus these three factors may be used as the basic factor set. As both the permissible quarter-replicate designs have BCD as a defining contrast, neither can be generated if B, C and D are used as basic factors.

4. Fractional Replication and Confounding

The algorithm, as presented in Section 2, is not suitable for producing designs which involve both fractional replication and confounding. Clearly an algorithm for generating such designs in a single pass would require specification of effects which were eligible for selection as defining contrasts and effects which were eligible for selection as confounded effects. However, repeated application of our algorithm will produce such designs.

Consider a design for a 2^n experiment in 2^{n-m} plots arranged in 2^r blocks of size 2^{n-m-r}. For such a design 2^m-1 defining contrasts and $2^m(2^r-1)$ confounded effects are required. These can be found by applying the algorithm twice; firstly to find a set of m defining contrasts to form a basis for the 2^m-1 defining contrasts and then to extend this set by r further effects so that these $m+r$ effects form a basis for the $2^{m+r}-1$ defining contrasts and confounded effects.

The extended procedure is as follows.

Run 1. Apply the algorithm in order to find a suitable group of defining contrasts. Record each suitable group and the $n-m$ basic factors from which the group was generated. Select one of these designs.

Run 2. Rewrite all the effects which are ineligible for confounding with blocks in terms of the $n-m$ factors which were the selected basic factors, i.e. replace each ineligible effect by that basic effect which it has as an alias. Use this set of rewritten effects as the ineligible set, eliminate the m added factors and reapply the algorithm with $n' = n-m$ factors and $m' = r$ added factors.

The justification of this process is simple. For a 2^n experiment in 2^{n-m} plots there are 2^{n-m} estimable effects, each of which has 2^m effects as aliases. However, no basic effect has another as an alias and as there are 2^{n-m} basic effects all other effects have exactly one basic effect as an alias. If an effect which is not a basic effect is ineligible for confounding with blocks then so is the basic effect which it has as an alias. As each basic effect has 2^m-1 other effects as aliases it is necessary only to search for 2^r-1 suitable confounded effects among the eligible basic effects. This search may be performed by reapplying the algorithm to the basic factors only.

As an example we look at a design given by Patterson (1965). This is a one-quarter replicate of a $4 \times 2^2 \times 4$ factorial design effectively in eight blocks of two plots each. This example may be expressed as a 2^{n-m} factorial with $n = 6$, $m = 2$, $r = 3$ and treatment factors A, B, C, D, E, F, where the interactions AB and EF denote main effects of the original 4-level treatment factors. Patterson (1965) applied the following conditions:

(1) all combinations of the levels of A, B, C and D, of A, B, E and F, of C, D, E and F must be included;

(2) the main effect of D must be confounded with blocks but no other main effect except AB, EF may be;
(3) the main effects A, E must not have two-factor interactions as aliases unless they include EF, AB respectively.

Ignoring blocking, the first condition can be satisfied by using the requirements set $\{A, B, AB, C, D, CD, E, F, EF\}$. The first run of the algorithm produces thirty-six quarter-replicate designs, none of which can be arranged in eight blocks of two plots without confounding some member of the requirements set. The second condition means that the effects ineligible for confounding with blocks are A, B, C, E, F and their interactions with D. Eight designs satisfy the first two conditions and two designs satisfy all three conditions. For these two designs we may write the $m+r$ effects as
(i) defining contrasts $BDF, ABCE$; confounded effects D, AC, AB;
(ii) defining contrasts $BDF, ABCDE$; confounded effects D, AC, AB.
These are the two designs obtained by Patterson (1965) using the design key method.

5. Generalizations

There are two obvious ways in which the results of this paper can be generalized. Firstly, the process described in Section 4, which relates to a fractional design with confounding, can clearly be extended to split plot fractions having confounding with blocks and further confounding with whole plots; and so on. Secondly, we have restricted our attention to 2^n designs but the results may be generalized to all p^n designs with p prime, as will be shown in a subsequent paper.

References

Berger, P. D. (1972). On Yates' order in fractional factorial designs. *Technometrics*, **14**, 971–972.
Cochran, W. G. and Cox, G. M. (1957). *Experimental Designs*, 2nd edn. New York: Wiley.
Das, M. N. (1964). A somewhat alternative approach for construction of symmetrical factorial designs and obtaining maximum number of factors. *Calcutta Statist. Ass. Bull.*, **13**, 1–17.
Greenfield, A. A. (1976). Selection of defining contrasts in two-level experiments. *Appl. Statist.*, **25**, 64–67.
Patterson, H. D. (1965). The factorial combination of treatments in rotation experiments. *J. Agric. Sci.*, **65**, 171–182.
—— (1976). Generation of factorial designs. *J. R. Statist. Soc.* B, **38**, 175–179.

Appendix A3.6

Algorithm for selecting defining contrasts in p^n experiments

In this appendix the basic algorithm outlined in §3.5.3 is described in detail.

The Algorithm

Step 1. Set the minimum m_1 and maximum m_2 values for m and set $m = m_2$.

Step 2. Choose a set of $n-m$ basic factors.

Step 3. Form a $p^{n-m} \times m$ table, the rows labelled by the basic effects, the columns by the added factors. Enter those products $B_i A_j$ that are eligible.

Step 4. Set $j = 0$ and defining contrast group $G(0) = I$.

Step 5. Increase j by 1 ; set $i(j) = 0$.

Step 6. If $i(j) = p^{n-m}$ then go to Step 10; otherwise increase $i(j)$ by 1 .

Step 7. Check whether all generalized products $B_{i(j)} A_j D_k$, as D_k ranges over all members of the defining contrast group $G(j-1)$, are eligible: if not, then go to Step 6.

Step 8. Extend $G(j-1)$ to $G(j)$ by including the effects $(B_{i(j)} A_j)^r D_k$ for $r = 1,\ldots,p-1$, as D_k ranges over all members of $G(j-1)$.

Step 9. If $j < m$ then go to Step 5; otherwise $G(m)$ is a group of eligible defining contrasts: take appropriate action and return to Step 6.

Step 10. If $j > 1$ then decrease j by 1 and return to Step 6.

Step 11. If there are new sets of $n-m$ basic factors available then select one and go to Step 3.

Step 12. If $m = m_1$ or if a suitable design has been found then stop; otherwise decrease m by 1 and return to step 2.

Comments on the algorithm

The algorithm will normally be used with a fixed value for m but m can be determined from the order of the largest group of eligible defining contrasts. Clearly $m \leq n$ and so the starting value for m could be set to n though a smaller value is normally preferred. The possible values for m are specified in Step 1 and the value of m is changed in Step 12.

Steps 2 and 3 are self-explanatory. The basic effects are listed as B_i, the added factors as A_j. Note that all powers of basic effects are listed. Step 4 initialises a search and Steps 5 to 10 describe the search for m defining contrasts, one from each column, that generate a group of eligible defining contrasts. The columns are labelled by j, the rows within each column by $i(j)$. The group generated by the defining contrasts chosen from the first j columns is denoted by $G(j)$. It contains p^j elements, as we include all the $p-1$ different ways of writing any treatment effect. (The search defined by Steps 5 to 10 is effectively a recursive process and the steps may be replaced by a definition in terms of recursion.)

Steps 5 and 6 are used to control the search process. Step 7 is used to check whether an effect in column j may be adjoined to the defining contrasts group $G(j-1)$ to form $G(j)$. The effect $B_{i(j)}A_j$ may be chosen as a defining contrast only if all the elements $(B_{i(j)}A_j)^r D_k$, for $1 \leq r \leq p-1$ and each element $D_k \in G(j-1)$, are eligible but it is necessary to test the eligibility only of the elements $B_{i(j)}A_jD_k$ as in Step 7. To demonstrate this we recall firstly that if an effect is ineligible than so are all powers of that

effect and secondly that for any value $r \neq 0 \mod p$ there is a value $s \mod p$ such that $rs = 1 \mod p$. It follows that if $(B_{i(j)}A_j)^r D_k$ is ineligible then so is its sth power, viz $B_{i(j)}A_j D_k^s$. Now $D_k^s \in G(j-1)$ and the result follows. If the effect $B_{i(j)}A_j$ is chosen as the jth defining contrasts then in forming G(j) from G(j-1) r must take all the values 1,2,...,p-1 as specified in Step 8. Step 9 is used to check whether a design has been found and if so to take appropriate action e.g. store and continue, or print and stop. Step 10 is used to check if the search has been completed.

At the commencement of Step 11 the algorithm has produced all groups of eligible defining contrasts that can be generated by m defining contrasts of the form $B_{i(j)}A_j$ for the given choice of n-m basic factors. As shown in Appendix 3.5 for $p = 2$ the algorithm generates all groups of defining contrasts containing no basic effects, and thus all possible groups if no basic effect is eligible. Since the integers modulo p form a field when p is prime, a slight modification to this argument shows that this is true for any prime p. If there are any eligible basic effects then a further step is required. Step 11 tests whether another set of basic factors is required and, if so, selects one and transfers control to Step 4.

Appendix A3.6.1

Simple modifications to the algorithm

The algorithm described in Appendix A3.6 is in its most general form. In practice a number of modifications may be made.

Initial choice of m Normally the value of m is fixed in advance but the algorithm can look for the largest group of defining contrasts. Too large a starting value for m results in wasteful searching which can be avoided by the inclusion of an upper bound. One such upper bound is the value k such that the largest group of ineligible effects has order $\leq p^{n-k}$. This bound is readily justified for if G is a group of eligible effects of order p^m and H is a group of ineligible effects of order p^r then G and H have only I in common so $p^m p^r$ divides p^n i.e. $m+r \leq n$ and thus $m \leq k$ if $r = n-k$. Sub-groups among ineligible effects normally consist of all interactions among a set of treatment factors. In practice these are easy to find and it is not critical if the largest group is not found.

Greenfield (1976) gives another useful way of obtaining an upper bound for m. This also may be used to help the choice of m_1 in Step 1. Suppose there are u effects in the requirements set: then information on u(p-1) degrees of freedom is required from the design, thus the inequality $u(p-1) \leq p^{n-m}-1$ must hold for a fraction, and $u(p-1) \leq p^n-p^m$ must hold for a single replicate block design.

Choice of basic factors If there is any set of n-m factors such that no effect involving only those factors is eligible, then it may be chosen to be the set of basic factors in Step 2 and Step 11 omitted. The algorithm generates all groups of eligible defining contrasts containing no basic effects: since with this set of basic factors no

basic effect is eligible, the algorithm generates all groups of eligible defining contrasts without needing to use a new set of basic factors.

Fractions The savings described here apply to conditions which are usually satisfied if a fractional replicate is sought, and may be in other cases.

a) If all main effects and components of two factor interactions are ineligible the rows labelled by basic factors in Step 3 may be omitted. In general, if a resolution N fraction is sought all rows labelled by basic effects involving fewer than N-1 factors may be omitted.

b) If the component effects of a given interaction are either all eligible or all ineligible and if all main effects and two factor interactions are ineligible, then in the table formed in Step 3, only one row of the p-1 labelled B_i, B_i^2, ..., B_i^{p-1} need appear, for any basic effect B_i. For if B_iA_1 and $B_i^rA_2$ are defining contrasts, then so is $B_i^{1-rs}A_1A_2^{p-s}$ for any integer s : choosing s so that rs = 1 modulo p gives the component $A_1A_2^{p-s}$ of a two factor interaction as a defining contrast. Thus only one row of those labelled B_i, B_i^2, ..., B_i^{p-1} is required and it is immaterial which is chosen, for a design with defining contrast $B_i^rA_j=B_iA_j^s$ differs from one with defining contrast B_iA_j only by a relabelling of the levels of A_j, and such a relabelling is irrelevant if all interactions involving A_j are either completely eligible or completely ineligible.

c) If all main effects and two factor interactions are ineligible then if $B_{i(j)}A_j$ is chosen as a defining contrast all other effects in the rows labelled by $B_{i(j)}$ and its powers may be declared

temporarily ineligible. The choice of another defining contrast in these rows would lead to a component of a two factor interaction in the added factors being a defining contrast.

Appendix A3.6.2

Extension to other block structures

The algorithm described in Appendices A3.6 and A3.6.1 is suitable for block structures with two block categories, one nested within the other. In this appendix we extend the use of the algorithm to hierarchical structures with more than two categories. We also indicate how the algorithm may be used with crossed block structures.

In the two-category case a group of defining contrasts G_1 is formed all of which are eligible for confounding with blocks. Among the remainder $n-m$ can be found to generate a group G_2 of order p^{n-m} such that the direct product $G_1 \times G_2$ is H, the complete group of defining contrasts. For b nested block categories, H is partitioned into b sub-groups $G_1, G_2 \ldots G_b$ with only the identity in common such that $H = G_1 \times G_2 \times \ldots G_b$. Eligibility conditions are specified for the first $b-1$ strata (all effects being eligible in the lowest stratum) and each group $H_k = G_1 \times G_2 \times \ldots G_k$ $(k \leq b)$ satisfies the conditions for the kth stratum.

The following procedure helps to shorten the search. G_1 is formed by a straightforward application of the algorithm described in Appendix A3.6 using as ineligible effects those ineligible at the first stratum. Before the application of the second stage the added factors are removed. Wherever added factor A_j occurs in the ineligible effects for subsequent strata it is replaced by the unique element $B_{i(j)}^{p-1}$ where $B_{i(j)}A_j$ was the defining contrast chosen from column j. The second stage is a reapplication of the algorithm to the remaining factors, using as ineligible effects the (possibly rewritten) effects which are ineligible at that stage. For each subsequent stage the current added factors are removed, the ineligible

effects rewritten and the algorithm reapplied. The justification is a straightforward extension of that given in Appendix A3.5 (p.325).

We illustrate the procedure for designs with crossed block structure. Consider a design with block structure

$$(X * Y)/Z$$

This design has four block strata - X,Y,X.Y and X.Y.Z . There are more strata than categories but the effects confounded in stratum X.Y are determined by those confounded in strata X and Y . As with the hierarchical structure, sub-groups of defining contrasts G_1,G_2 and G_3 are formed corresponding to the three block categories. However G_2 must satisfy the eligibility conditions placed on stratum Y while $G_1 \times G_2$ must satisfy the conditions placed on stratum X.Y. The procedure then follows that for nested block structures but the suggested improvement of eliminating the added factors at each stage is no longer valid.

Appendix A3.6.3

Using a single set of basic factors

In this appendix the selection of sets of basic factors is considered for the case of two nested block factors. If there is no basic effects group containing no eligible effects and n is large it is inefficient to try all $\binom{n}{m}$ possible sets of basic factors. The more direct method given here has the advantages of avoiding wasteful searching and not repeating designs.

Choose a set of basic factors with few eligible basic effects. Run the algorithm with this set; only those groups of eligible defining contrasts containing one or more eligible basic effects are not produced. If the eligible basic effects are denoted E_1 , E_2 ... E_u then a sequence for generating these groups is defined as follows:

For $s = 1, 2 \ldots u$ generate all groups of confounded effects containing E_s but not E_t for any $t < s$.

This sequence may be readily achieved by generating all groups containing E_1 , declaring E_1 to be ineligible then repeating the process for E_2 etc. Let E be one of the effects E_1, E_2 ... E_u then a procedure is required for generating all groups containing E . The procedure given below is a modification of the one described in the previous section for multiple nested strata: effectively the procedure involves considering the current stratum as two nested strata with only effect E eligible in the first stratum.

All the factors occurring in effect E are basic factors. Let B be one of these factors then for some $r \neq 0$ modulo p the element E^rB does not contain B . Remove factor B and replace it by E^rB whenever it occurs in the ineligible effects. Reapply the algorithm with the remaining $n-1$ factors to obtain groups of order

p^{n-m-1} : adjoining E to these groups produces groups of eligible confounded effects of order p^{n-m}. If at the next stage there are no eligible basic effects then all groups of order p^{n-m-1} are generated, otherwise it is necessary to repeat the procedure for eligible basic effects E'_1, E'_2 ... E'_v. This recursive process can be achieved by modifying the algorithm at Step 11, so that for each eligible basic effect the ineligible effects are redefined as above and the algorithm called recursively for one fewer factor.

CHAPTER 4

OTHER CONFOUNDED AND FRACTIONAL FACTORIAL DESIGNS

4.1 Summary

In this chapter extensions to topics of Chapter 3 are discussed. These include general asymmetric factorial experiments, orthogonal and balanced fractions of 2^n and 3^n designs and some aspects of response surface designs. The constructions are interlinked; e.g. designs for 3^n experiments can be used to construct asymmetric fractional factorial and response surface designs. Optimal design procedures impinge on this chapter but discussion is deferred to Chapter 8.

In §4.2 literature relevant to the construction of confounded and fractional asymmetric factorial designs is reviewed. Confounded designs are discussed in §4.3; particular attention is given to the extensions of the design key and the generalized cyclic methods and to deletion of factor levels in symmetric designs. In §4.4 we consider fractions of 2^n and 3^n designs which are more general than those discussed in Chapter 3; simple generators are described for orthogonal and balanced 2^n designs and for 3^n designs. Construction of fractional asymmetric factorial experiments is discussed in §4.5 with particular reference to orthogonal main effect designs. In §4.6 we consider briefly the construction of response surface designs.

4.2 Review of the literature

Confounded designs. Yates (1937a) presents constructions for confounded 2^n3 experiments balanced in three superblocks with blocks of $3 \times 2^{n-1}$ or $3 \times 2^{n-2}$ units. He also indicates general methods for balanced 2×3^n designs. Binet et al. (1955) provide a selection

of single replicate designs and an analysis appropriate to quantitative factor levels. The designs they present are an ad hoc collection, many based on a single superblock from designs of Yates (1937a) or Li (1944). Several papers on confounded asymmetric factorial experiments were written by Indian authors during the period 1945-1965 (e.g. Shah (1958)) but most were concerned with balanced designs which had many replicates. Kishen and Srivastava (1959) present useful constructions based on deleting factor levels.

A somewhat different approach is adopted by Kramer and coworkers (Kramer and Bradley (1957), Brenna and Kramer (1961)) who apply factorial experiments to PBIB designs. Kramer and Bradley (1957) attribute to Bose useful remarks on the difference between the use of PBIB designs for factorial and variety trials:

1) In variety trials all treatment contrasts tend to be of roughly equal importance but in factorial experiments PBIB(2) designs with widely different λ_1 and λ_2 may be useful.

2) Some group divisible designs are disconnected. They are not useful in variety trials but may be in factorial experiments.

Brenna and Kramer (1961) point out that ϕ-series designs (§2.6.2) with $k < s - 1$ though not used extensively in variety trials are useful for factorial experiments and they provide a construction related to the Shrikande method for rectangular lattices. John and Smith (1972) consider the general problem of assigning an $m \times n$ factorial experiment to proper incomplete block designs and derive sufficient conditions for the design to have orthogonal factorial structure. (A design is said to have orthogonal factorial structure if the adjusted treatment sums of squares can be partitioned orthogonally for main effects and interactions.) Cotter, John and Smith (1973) extend the conditions to multi-factorial experiments and show that type A

designs of Kurkjian and Zelen (1963) have factorial structure.

White and Hulquist (1965) present a method for constructing single replicate or fractional asymmetric factorial experiments when the number of levels of all factors is prime. They define addition and multiplication of elements from distinct finite fields by mapping elements into a finite commutative ring containing sub-rings isomorphic to each field. Thereafter, the methods of construction are similar to those for finite fields. Raktoe and Federer (1972) consider other methods of combining elements from distinct fields but are unable to generalize the work of White and Hulquist.

John (1973) presents a generalized cyclic construction and shows that the designs have orthogonal factorial structure. He demonstrates via an example that designs with the highest mean efficiencies do not necessarily give the highest efficiencies for main effects. John (1979) shows that for generalized cyclic designs the efficiency of an interaction can be determined by restricting attention to the s factors involved in the interaction.

Dean and John (1975) give a generalized cyclic construction for single replicate designs and present initial blocks for some efficient designs. Patterson (1976) points out that treatments in the initial blocks of these designs form subgroups of the treatment group, whereas the extension of the design key method to asymmetric designs yields a similar procedure with a wider definition for the initial block. Designs constructable by the latter method only are often the most interesting. Dean and Lewis (1980) further extend the definition for the initial block.

Bailey, Gilchrist and Patterson (1977) generalize the method of White and Hulquist (1965) for combining elements from distinct fields. They also generalize the definition of treatment effects and give an

orthogonal breakdown of treatment effects applicable to asymmetric factorial designs. They then extend the design key method of relating plot and treatment effects to asymmetric designs. The class of designs covered is essentially the same as that of Dean and John (1975). Bailey (1978) extends the method to simple block structures; Patterson and Bailey (1978) give a simple introduction. The design key procedure and the generalized cyclic procedure as given by Dean and Lewis (1980) are closely related, but each can do certain jobs which the other cannot.

Fractional symmetric designs: Plackett and Burman (1946) show that a resolution III two-level factorial design with orthogonal main effects can be constructed for $4n-1$ treatments in $4n$ plots if there exists a Hadamard matrix of order $4n$. They provide simple procedures for constructing Hadamard matrices of all possible orders $4n \leq 100$ except $4n = 92$. (For a general review of the properties and applications of Hadamard matrices see Hedayat and Wallis (1978)). Plackett and Burman (1946) also construct resolution III designs for general p^n factorial experiments by the standard methods and demonstrate that they may be constructed also by a cyclic procedure.

S.R. Webb, reported by Margolin (1969a), modifies Box and Hunter's (1961) definitions of the resolution of a design to make them more appropriate when effects are not orthogonal. The definition is as follows:

A design is of resolution III if it is necessary to assume that some or all of the two-factor interactions are negligible for the main effects to be estimable. A design is of resolution IV if all main effects are estimable but not all the two-factor interactions are estimable. A design is of resolution V if all main effects and two-factor interactions are estimable.

Srivastava and colleagues in a series of papers reviewed by Srivastava (1978) construct non-orthogonal 'optimal balanced' 2^n designs of resolution V for $n = 5,6,7,8$. The definition of a balanced design is restrictive and refers to designs for which the variance-covariance matrix of the treatment effects is unchanged by permuting factors but not interchanging levels. The definition of optimality is also restricted, referring only to designs in the class and the selection of optimal balanced designs is akin to the selection of efficient cyclic designs. Srivastava (1978, p.271) notes that many designs for which the number of units approaches the lower bound have low efficiency.

Fractional 3^n designs based on a selection of points lying on particular hyperspheres around the mid-point of designs have been used by several authors; each factor has levels -1, 0 or 1 and any treatment with r^2 factors having non-zero levels lies on a hypersphere of radius r . Debaun (1959) considers the selection of geometrical configurations (which correspond to the spheres) from a 3^n design. Box and Behnken (1960) compound a 2^k factorial design with a BIB design for n treatments in blocks of k units; if the factorial design is a single replicate and the BIB design is unreduced then the resulting design contains all points on the hypersphere of radius $\sqrt{k}$ but otherwise it contains a fraction of these points. Adding an appropriate number of centre points makes the design rotatable under a second-order model. Fry (1961) presents a construction which corresponds to the selection of the points on alternate hyperspheres. He is concerned with classical factorial analyses and adds a centre point to make all levels of each main effect equally represented. Hoke (1974) constructs economical second-order designs by considering combinations of fractions of points taken from

hyperspheres of radius r where $0 \leq r^2 \leq 3$ and $n-3 \leq r^2 \leq n$. Anderson and Thomas (1979) give two constructions for resolution IV designs where the number of units approaches the lower bound given by Margolin (1969a); the second construction is very similar to those of Hoke (1974). As with the 2^n designs with near minimal number of points, the mean efficiency of these designs is low, little more than 0.5.

Fractional asymmetric designs: Confounded single replicate designs may be used to form fractional designs by selecting some but not all blocks. In particular the design key and generalized cyclic generators are readily adapted to fractional designs as is the method of deleting factor levels. Connor and Young (1961) combine fractions of 2^n and 3^m designs to form a useful catalogue of fractional 2^n3^m designs; the designs may also be constructed by design key generators.

Although many of the designs of Connor and Young have good orthogonality properties with regard to main effects and low order interactions, there is often difficulty in obtaining orthogonality in fractional designs unless unequal replication is permitted. Plackett (1946) gives the conditions for orthogonality of effects and shows how to construct an orthogonal main effects design for a fractional 3^5 design in 16 units by collapsing one level of a 4^5 factorial. Addelman and Kempthorne (1961) and Addelman (1962b) use this technique to construct numerous orthogonal main effects designs. Margolin (1969b) shows that collapsing levels in a 3^{n+m} factorial does not reduce the resolution and under certain conditions can increase it. He also shows how level collapsing can be used with Connor and Young's (1961) construction for fractional designs to give resolution V designs. Pesotan, Raktoe and Worthley (1978) obtain

unequal replication though a generalized fold-over principle where level i of a t level factor is folded to level $t-1-i$ but the designs presented seem of little practical interest.

Lewis and John (1976) criticise the use of orthogonal main effects designs obtained by unequal replication on the grounds that the wrong hypothesis about main effects is tested. This is a valid complaint; another is the loss of efficiency introduced by unequal replication. Orthogonal main effects designs should therefore be used with caution.

Response surface designs: This class of designs is too extensive to review in detail here. Useful descriptions are to be found in John (1971), Raghavarao (1971) and Hill and Hunter (1966).

4.3 Confounded Designs

In this section some useful constructions for confounded asymmetric factorial designs are described in more detail. We restrict attention to methods giving few replications for we think that with three or more treatment factors the introduction of further factors is generally more useful than achieving a high degree of replication.

4.3.1 Deleting factor levels

The confounded asymmetric factorial designs given by Yates (1937a) can be constructed by deleting levels in confounded symmetric designs. A balanced 2^n3 design in two blocks of $3 \times 2^{n-1}$ units can be derived by deleting one level of the four level factor in a balanced 2^n4 design with interactions ABCD' , ABCD" , ABCD'" confounded in successive superblocks. This design in turn may be constructed from a symmetric 2^{n+2} design. Balanced designs for 2^n3 treatments in blocks of $3 \times 2^{n-2}$ can be derived from a 2^n4 design by

confounding in the three superblocks A'U, A"V, A"'UV; A"U, A"'V, A'UV; A"'U, A'V, A"UV respectively where U and V are interactions among two-level factors and A is the four-level factor. The $2^{\frac{1}{3}}3^{m}$ designs are constructed from 3^{m+1} designs by deleting a level of factor A where ABC^2, AB^2C, ABC and AB^2C^2 are confounded in successive replicates.

A justification for deleting levels (and, incidentally, for collapsing levels) in a symmetric factorial design is given by Kishen and Srivastava (1959). They show that any function of the elements of a field of order n can be represented as a polynomial

$$f(x) = \Sigma a_i x^i$$

where a_i are elements of the field. Collapsing factor levels may therefore be represented as the result of applying a polynomial function and deleting levels regarded as the removal of repeated levels.

Let $s=p^u$ then levels can be deleted from any set of k treatment factors if no interaction among the k factors is confounded with blocks. Kishen and Srivastava construct the following balanced designs:

i) A ts factorial in s blocks of t units ($t < s$): balanced in s-1 superblocks. These designs, which correspond to ϕ series designs are formed by confounding AB, AB^2... in successive superblocks and deleting levels of A.

ii) A ts^2 factorial in s blocks of st units ($t < s$): balanced in s-1 superblocks. Confound the effects ABC, AB^2C^2, AB^3C^3... etc. in superblocks then delete levels of A.

iii) A uts factorial in s blocks of tu units ($t,u < s$): balanced in $(s-1)^2$ superblocks. Confound the effects A^iB^jC ($1 \leq i,j \leq s$) in successive superblocks then delete levels of A and B.

Confounded designs formed by deleting treatment factor levels have s^c blocks per superblock and $d(\geq c \geq 1)$ factors with s levels.

4.3.2 Generalized cyclic constructions

Generalized cyclic generators, used in §2.4 to construct block designs, can be used also to construct confounded asymmetric factorial designs. Consider a factorial experiment with m treatment factors, the ith factor having t_i levels and $t = t_1 t_2 \ldots t_m$ treatments in all. (In this chapter we do not distinguish factors from pseudo-factors.) Represent treatments by m-tuples $\underline{a} = a_1 a_2 \ldots a_m$ where $0 \leq a_i \leq t_i - 1$ then addition of treatments is as defined in §2.4.2. Under this operation the treatments form an abelian group H of order t. A full GC/m design with t blocks is constructed by selecting an initial block of k treatments and forming the ith block by adding the ith member of H to each treatment in the initial block.

Each treatment generates a cyclic sub-group of order $\frac{\gamma}{w}$ (§2.4.2). We denote by $A \oplus B$ the direct sum of two sets with no common element other than identity (§2.4.2). Construction of treatments by generators corresponds to forming group H from the direct sum of cyclic sub-groups. A subset of generators form a normal sub-group H_0, say, of H. The remaining generators also form a normal sub-group H_1 and H is the direct sum of H_0 and H_1

$$H = H_0 \oplus H_1 .$$

We now state as a theorem two constructions of fundamental importance in §§4.3.2 and 4.3.3. The sufficiency of the conditions are demonstrated but they are also necessary; for a rigorous proof see Dean and Lewis (1980).

Theorem 4.1 i) Let the initial block be the direct sum $H_0 \oplus S$ of the sub-group H_0 of order h_0 and the set S of order s containing the identity. If H

is ordered so that the $(h_1 i+j)$th element is the sum of the ith element of H_o and the jth element of H_1 where $h_1 = t/h_o$ is the order of H_1 $(o \leq i < h_o;\ o \leq j < h_1)$ then the first h_1 blocks of the generalized cyclic design form s complete replicates and the remaining blocks are repeats of those.

ii) If also, $H_1 = S \oplus R$ where R is a set of order $r = h_1/s$ containing the identity and H_1 is ordered so that the $(ri+j)$th element is the sum of the ith element of S and the jth element of $R(o \leq i < s;\ o \leq j < r)$ then the design is resolvable.

The following example illustrates the theorem.

Example 4.1 Let $t_1 = 3$, $t_2 = 2$ and $t_3 = 2$ and

$$H_o = \{000,\ 011\}$$

$$S = \{000,\ 101,\ 200\}$$

$$R = \{000,\ 010\}\ .$$

The values of t, h_o, s, h_1, r are 12, 2, 3, 6 and 2 respectively. The principal block and block generators are

$$H_o \oplus S = \{000,\ 011,\ 101,\ 110,\ 200,\ 211\}$$

$$H_1 = S \oplus R = \{000,\ 010,\ 101,\ 111,\ 200,\ 210\}$$

The first $h_1 = 6$ blocks of the full design are given by columns:

000	010	101	111	200	210
011	001	110	100	211	201
101	111	200	210	001	011
110	100	211	201	010	000
200	210	001	011	100	110
211	201	010	000	111	101

It is readily verified that addition of treatment 011 to all treatments leaves the blocks unchanged and that the first r=2 blocks form a superblock. (Observe also that the first $h_o = 2$ rows form a superblock.)

The sufficiency of i) is seen by noting that the initial block contains the sub-group H_o and $s-1$ cosets formed by adding the elements of S to H_o where $S \subset H_1$. The first h_1 blocks contain the direct sum of each of the s sets with H_1 but in all cases the direct sum is the group H so the first h_1 blocks contain s replicates. All other blocks are formed by addition of an element from H_o to one of the first h_1 blocks but such addition leaves the block unchanged. The first h_1 blocks are repeated h_o times in the full design and are called a $1/h_o$ fraction of the full design. If $H_o \oplus S = H'_o \oplus S'$ for some group H'_o with order $h'_o > h_o$ then a smaller fraction can be constructed based on H'_o.

The sufficiency of ii) is seen by noting that the initial r blocks contain all treatments in the set $(H_o \oplus S) \oplus R$ but

$$\begin{aligned}(H_o \oplus S) \oplus R &= H_o \oplus (S \oplus R) \\ &= H_o \oplus H_1 \\ &= H\end{aligned}$$

and so the blocks contain a complete replicate.

Dean and John (1975) give a list of generators for single replicate confounded designs with the initial block forming a sub-group. Dean and Lewis (1980) give examples of using the more general generator for constructing both confounded factorial designs and block designs.

4.3.3 Design key constructions

Bailey, Gilchrist and Patterson (1977) extend the definition (§3.3) of a treatment effect as follows: a treatment effect $T(\underline{x})$ is the set of all contrasts between treatments $\underline{z}$ with different values of $[\underline{z},\underline{x}]$ where

$$[\underline{z},\underline{x}] = \underline{q}^T \Delta_t \underline{x} \mod \gamma_t \qquad 1)$$

γ_t is the lowest common multiple of the t_i and Δ_t is the $n \times n$

diagonal matrix with element (i,i) given by γ_t/t_i . For symmetric designs Δ_t is a unit matrix and the definition 1) reverts to that in §3.3. (It is, however, applicable to the case where t_i is not prime power.) The definition of plot effects is extended in a similar manner. Bailey et al. identify plot and treatment effects by the linear relation

$$\Delta_p \underline{y} = K(\Delta_t \underline{x}) \qquad 2)$$
$$\underline{y} = K_* \underline{x}$$

where the jth element of y is reduced modulo p_j and $K_* = \Delta_p^{-1} K \Delta_t$. Both K and K_* are integer matrices. The design is constructed by allocating treatment $\underline{q}$ to plot $\underline{r}$ where,

$$\underline{q} = K^T \underline{r} \qquad 3)$$

and q_i is reduced modulo t_i .

Example 4.2 The design in Example 4.1 can be constructed by a design key generator as follows:

Let T_1 be the three-level treatment factor and T_2, T_3 be the two-level factors. Define the six blocks by two pseudo-factors P_1 and P_2 with three and two levels respectively and the six plots per block by pseudo-factors P_3 and P_4 with three and two levels respectively. Represent each unit by the vector of block pseudo-factor levels $(P_1, P_2, P_3, P_4)^T$ then the design key matrix

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

yields the design. The fourth unit in the third block, for example, corresponds to the vector (1, 0, 1, 1) and has treatment (2, 1, 1) .

Inversion of matrix K_* to determine confounded effects can

cause problems and Bailey *et al.* (1977) suggest finding solutions for $\underline{x}$ from the vectors $\underline{y}$ by trying each vector in turn. For fractional designs the defining contrasts of the fraction are given by $\underline{x}_o$ satisfying $K_*\underline{x}_o = 0$; the vectors $\underline{x}_o$ form a sub-group D.

The restriction that both K and K_* are integer matrices implies that the element (j,i) of the key matrix K satisfies the condition

$$k^*_{ji} = k_{ji} \frac{p_j}{t_i} \cdot \frac{\gamma_t}{\gamma_p} \qquad 4)$$

is also integer. Commonly, $\gamma_t = \gamma_p$ and therefore $k_{ij}p_i/t_j$ is to be integer. If all treatment and block factors have prime number of levels and $p_j = t_i$ then any value k_{ji}, $o \leq k_{ji} \leq t_i - 1$ satisfies the condition but if $p_j \neq t_i$ then k_{ji} must be a multiple of t_i. Reducing k_{ji} modulo t_i, however, leaves the design unchanged and therefore all elements (j,i) where $p_j \neq t_i$ can be replaced by zero. This is the basis of the construction implicitly used by Patterson and Bailey (1978).

The effects $T(\underline{x})$ are not orthogonal for asymmetric designs. Bailey *et al.* (1977) provide a neat way of defining orthogonal effects $T_*(\underline{x})$ based on consideration of cyclic sub-groups. The essential relation in this construction is

$$[\underline{z}, r\underline{x}] = r[\underline{z}, \underline{x}] \mod \gamma_t \qquad 5)$$

and hence if $\underline{x}$ forms a cyclic sub-group of order s then $[\underline{z},\underline{x}]$ takes s different values over this set and corresponds to $s-1$ degrees of freedom. The $T_*(\underline{x})$ are defined for main effects, for two-factor interactions they are defined as those effects orthogonal to main effects etc. For given $T(\underline{x})$ confounded with blocks the appropriate $T_*(\underline{x})$ can be determined. Orthogonal effects $T_*(\underline{x})$ appear to be of most use in the class of designs covered by 1) and

the construction of Dean and John (1975) but formation of the $T_*(\underline{x})$ is readily performed by a small computer program.

Examples of the use of design key generators are given by Patterson (1976), Bailey et al. (1977) and Bailey (1977).

4.3.4 A comparison of generalized cyclic and design key generators

Theorem 4.1 (§4.3.2) is important in comparing generalized cyclic and design key generators given by Dean and John (1975) and Patterson (1976) (§4.3.3). In the construction given by Dean and John (1975) the set S is the zero element and the initial block is always a sub-group of H. In the design key method a wider selection of sets S is permitted but restrictions are caused by the nature of the construction.

Example 4.3 3×2^2 design, Plan 6.9 Cochran and Cox (1957).

The GC/3 generator given in Example 4.1 produces the design in the order Rep II, Rep III, Rep I. $H_o \oplus S$ is not a sub-group and the method of Dean and John (1975) cannot construct the design. The set S can be formed by the design key method and thus the matrix given in Example 4.2 generates the same design as the generalized cyclic generator.

Example 4.4 3×2^3 design, Plan 6.10 Cochran and Cox (1957).

Let $H_o = \{0000, 0111\}$

$S = \{0000, 1001, 2010\}$

$R = \{0000, 0001, 0010, 0011\}$

The GC/3 procedure generates the replicates in the order Rep I, Rep III, Rep II. $H_o \oplus S$ is not a sub-group and there are no sets S which may be generated by the design key method so neither the method of Dean and John (1975) nor design key method can be used to construct this design.

The generalized cyclic method is more general than the design key method for the simple block structures so far described but the latter permits more flexible use of incomplete cycles and also provides a compact procedure. Because both methods are useful and simple, they could both be held in a small computer program and for any given design the most appropriate method selected.

4.4 General fractions of 2^n and 3^n factorial experiments

In this section more general fractions of 2^n and 3^n factorial experiments than those described in Chapter 3 are discussed.

We note the main differences in the requirements for 3^n designs with qualitative and quantitative levels. For qualitative levels ordering is rarely assumed and there is little attempt to isolate individual degrees of freedom as information is usually required on all or no components of a given effect. Blocking is based on flats in Euclidean $E(n,3)$ space i.e. on the I and J type contrasts. Designs of resolution III, IV and V are determined by whether all main effects, interactions etc. are estimable. For quantitative levels ordering is important and the linear and quadratic contrasts are usually regarded as the most interesting. The levels are commonly represented as -1, 0, 1 and blocking based on hyperspheres about the centre-point of the design. The designs are often used for estimating second-order response surfaces in which case interactions involving quadratic effects are not required. The definition of type V designs can therefore be less restrictive for quantitative than for qualitative levels.

4.4.1 Orthogonal 2^n designs

Plackett and Burman (1946) construct orthogonal main effect designs for $4n-1$ factors in $4n$ units of resolution III from Hadamard

matrices; resolution IV designs for the $4n$ factors in $8n$ units can be formed by the fold-over technique. The resolution III design is formed by treating the matrix as a design for $4n$ factors in $4n$ units and ignoring the factor which occurs at one level only; the resolution IV design is obtained by repeating the basic design with levels reversed. Fuller descriptions are given by John (1971, Chapters 8, 9). The designs are closely related to a class of BIB and weighing designs, which may be constructed using Hadamard matrices; methods for generating small Hadamard matrices are therefore noted in Appendix A4.1.

4.4.2 Balanced 2^n designs

Optimal 'balanced' 2^n designs may be constructed as follows:

i) Form a non-negative integer vector $\underline{w} = (w_o, w_1, \ldots, w_n)$.

ii) Replicate w_i times all treatments $\underline{x}$ with exactly i factors having level 1 $(o \leq i \leq n)$.

Optimal balanced designs can be obtained from sundry published tables or by the analytical approach of Srivastava (1978). These will normally suffice but non-optimal designs may be preferred in certain circumstances as, for example, when estimates of main effects need to be estimated more accurately than interactions. Determining the properties of rival designs is rarely arduous.

Example 4.5 A design for a 2^7 factorial in 48 units is required.

The number of treatments with $0, 1 \ldots 7$ factors at level 1 are respectively $1, 7, 21, 35, 35, 21, 7, 1$. The promising choices for the w_i (i.e. those for which $w_o + w_7 \leq 6$) are

a) $w_o + w_7 = 6, \quad w_1 + w_6 = 1, \quad w_3 + w_4 = 1$

b) $w_o + w_7 = 6, \quad w_2 + w_5 = 2$

c) $w_o + w_7 = 6, \quad w_1 + w_3 = 3, \quad w_2 + w_5 = 1$.

Consider the designs b). As 29 parameters are to be estimated at least this number of distinct points are required so $w_2 = 0$ or $w_5 = 0$ are not permitted and hence only the value of w_o can be changed. The design's properties are unaltered by interchange of all levels 0,1 so only the cases $w_o = 0,1,2,3$ need be considered. The determination of the latent roots of four 29×29 matrices is a straightforward task.

The optimal designs are easily generated from small arrays and are therefore readily included in a compact computer program. The optimal properties suggest that they provide a useful addition to the range of two-level factorial designs.

4.4.3 Fractional 3^n designs

The designs of Debaun (1959), Box and Behnken (1960), Fry (1961), Hoke (1974) and some of Anderson and Thomas (1979) can be constructed as special cases of a general procedure. Let each factor have levels -1, 0, 1 and the mid-point of the design be (0...0) . The general procedure is then

i) Form a non-negative vector $w = (w_o, w_1 \ldots w_n)$ whose ith element contains the replication for points on hypersphere with $r^2 = i$.

ii) To each hypersphere with non-zero w_i apply a routine to obtain a set of vectors $\{S_i\}$ with i units and n-i zero values.

iii) To each vector S_i apply a routine for generating vectors with the same pattern of non-zero elements but with signs applied.

The construction is readily justified by expanding 3^n :

$$3^n = (1+2)^n = \sum_{o}^{n} \binom{n}{i} 2^i .$$

The three steps correspond respectively to selecting from n+1 terms, $\binom{n}{i}$ vectors and 2^i allocations of sign.

The procedures mentioned above correspond as follows:

Debaun (1959):
1) variable weights w_i
2) select all vectors
3) select all combinations of sign.

Box and Behnken (1960):
1) w_o and w_k only are non-zero ($k \leq n$)
2) select according to BIB designs for n treatments in blocks of k units
3) apply signs as determined by a 2^{k-u} fractional factorial ($o \leq u \leq k$).

Fry (1961):
1) if i even, $w_i = 1$ else $w_i = 0$; $w_o = 2$
2) select all vectors
3) select all combinations of sign.

Hoke (1974):
1) variable w_i
2) select all vectors
3) if $i < n$ all signs are the same $i = n$ choose all $\binom{n}{j}$ ways of having exactly j positive.

Anderson and Thomas (1979):
1) $w_o, w_1, w_{n-1}, w_n = 1$; else $w_i = 0$
2) select all vectors
3) $i = 1, n-1, n$ all signs the same, plus: $i = n$ all ways of choosing one sign opposite to the rest.

Notes: 1) Choosing i out of n in all ways yields an unreduced BIB design. The Box and Behnken application may thus be regarded as the general case for step 2.

2) Das and Narasimham (1962) extended Box and Behnken's method by replacing values ± 1 of the ith hypersphere by suitably chosen

values $\pm C_i$; central composite designs can be constructed in this way.

Example 4.6 Starks (1964) presents an orthogonal main effects 3^7 design in 16 plots. An alternative construction is

1) $w_o = 2$, $w_4 = 1$

2) select according to BIB (7,4,4,7) , cyclic with initial block (0 3 5 6)

3) for jth block form both ways of element j having sign opposite to the rest.

The design is therefore: cyclically permute

$$\begin{matrix} & (-1 & 0 & 0 & 1 & 0 & 1 & 1) \\ & (1 & 0 & 0 & -1 & 0 & -1 & -1) \\ +2* & (0 & 0 & 0 & 0 & 0 & 0 & 0) \end{matrix}$$

The construction outlined in this section is quite simple and generates useful designs. It is similar to that of §4.4.2 and both could be included within a small computer program.

4.5 Fractional designs for asymmetric factorial experiments

In this section two widely used constructions for fractional asymmetric factorial experiments are described. The first, sometimes known as 'conjoining fractions' is due to Connor and Young (1961). It is closely related to the design key generator of Patterson and Bailey (1978) and the two topics are discussed together. The second, collapsing factor levels, is the principal method for forming orthogonal main effect plans; we discuss this and other methods.

4.5.1 Conjoining fractions

Connor and Young (1961) present a useful catalogue of fractional

$2^n 3^m$ designs for most values of $n+m \leq 10$. The designs are constructed as follows:

1) Form fractional 2^{n-u} and 3^{m-v} designs S_o and S'_o respectively

2) Form the design $\sum_i^k S_{n_i} \times S'_{m_i}$ where S_{n_i} and S'_{m_i} indicate the n_ith and m_ith coset of S_o and S'_o ($o \leq n_i \leq 2^u - 1$, $o \leq m_i \leq 3^v - 1$) and $S_{n_i} \times S'_{m_i}$ indicate the product formed by taking all combinations of treatments. The resulting design has $k2^{n-u}\,3^{m-v}$ units and is thus a $k2^{-u}\,3^{-v}$ fraction.

The designs are easily constructed using the special form of the design key operation described by Patterson and Bailey (1978). The design key matrix in every case satisfies $k_{ji} = o$ if $p_j \neq t_i$ (§4.3.3). The k component products $S_{n_i} \times S'_{m_i}$ are formed from the same set of defining contrasts and hence differ only in the choice of base vector (§3.3.1).

Example 4.7 Connor and Young (1961, p. 20) construct a ½ replicate of a $2^3 3^2$ factorial design by combining fractions based on defining contrasts $A_1A_2A_3$ and B_1B_2 respectively. The design is then formed $S_o \times S'_o + S_1 \times S'_1 + S_1 \times S'_2$ which is equivalent to $S_o \times S'_o + S_1 \times S'_1 + S_o \times S'_2$. Let plot factors P_2, P_2 and P_3 have 1, 2 and 3 levels respectively then the design key matrix required to generate $S_o \times S'_o$ is:

$$\begin{vmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \end{vmatrix} \qquad 1)$$

The base vectors yielding the components $S_o \times S'_o$, $S_1 \times S'_1$ and $S_o \times S'_2$ are:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \qquad 2)$$

But this construction is also given by design key matrix

$$\left[\begin{array}{ccccc} \overline{0} & 0 & 1 & 0 & \overline{1} \\ \hline 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ \underline{0} & 0 & 0 & 1 & \underline{2} \end{array}\right] \qquad 3)$$

The chief deficiency of the tables of Connor and Young (1961) are the absence of blocking schemes and the limited usefulness of the single design presented for some values of n and m. The use of design key generators can overcome these problems; e.g. the first plot factor for design key matrix 3) can be identified with blocks thereby yielding a design for three blocks of 12 treatments.

4.5.2 Orthogonal main effect designs

Orthogonal main effect designs are commonly regarded as designs in which only main effects are of interest. There is no need for this restriction and most of the constructions considered here allow higher order effects to be estimated. Several constructions are commonly useful.

i) conjoining fractions (§4.5.1)

ii) forming orthogonal arrays of strength 2

iii) replacing treatment factors by sub-factors and superfactors (§1.3)

iv) deleting factor levels (§4.3.1)

v) collapsing factor levels.

The constructions can be used in combination, the last three being commonly applied after the first two.

i) Conjoining fractions. The majority of designs presented by Connor and Young (1961) (see Examples 4.7, 4.8) have orthogonal main effects and allow interactions to be estimated.

ii) Orthogonal arrays of strength 2. This construction is most useful for symmetric factorial experiments when main effects plans which cannot be constructed with a design key generator are required. The most important class are the designs of Plackett and Burman (1946) (§4.4.1). Useful additions are a 3^7 design in 18 units (Bose and Bush (1952)) and designs for up to $2(s^n - 1)/(s - 1) - 1$ treatment factors with s levels in $2s^n$ units where s is prime or prime-power (Addelman and Kempthorne (1961)). These designs for $s \neq 2$ could be stored complete.

iii) Sub-factors and super-factors. Consider a fractional factorial design with orthogonal main effects which contains a subset of factors $\{T_i\}$ such that all effects involving only these factors are orthogonal to the main effects of all other factors, then on combining the factors T_i into a super-factor S, say, the resulting design also has orthogonal main effects.

Example 4.8 Let A_1, A_2, A_3 and B_1, B_2 be the three two-level factors and B_1, B_2 be the two three-level factors of the $2^3 3^2$ design of Example 4.7. All effects involving A_1, A_2, B_1 and B_2 are estimable and all except $A_1.A_2$ are orthogonal to A_3. Let factors A_1, B_1 be replaced by super-factor C and A_2, B_2 similarly by D then the design generated by the design key matrix 3) has orthogonal main effects:

000	120	210	011	101	221	020	110	200
301	421	511	310	400	520	321	411	501
031	151	241	040	130	250	051	141	231
330	450	540	341	431	551	350	440	530

Example 4.9: In orthogonal main-effects designs for $4n-1$ treatment factors in $4n$ units, two-level factors may not normally be replaced by four-level factors for interactions between pairs of factors are confounded with other main effects.

If, however, the underlying Hadamard matrix has the form $H_{4n} = H_m \otimes H_2$ where $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ then there exists one factor A_m such that A_{m+i} is aliased with $A_m.A_i$ $(i \leq m-1)$. Any three factors A_m, A_i and A_{m+i} may therefore be replaced by a single four-level factors giving a $2^{4n-4}.4$ orthogonal main effects design in $4n$ units.

Example 4.10 The 3^7 design in 18 units with orthogonal main effects given by Bose and Bush (1952) is resolvable and one factor A, say, is orthogonal to all others in each super-block. A two-level factor identified with super-block may be introduced to give a 2.3^7 design then combined with A to yield a six-level factor to give a $3^6.6$ design. (As in Example 4.9 all degrees of freedom are exhausted by the final design.)

iv) Deleting factor levels. This operation is normally applied to symmetric factorial designs but can also be applied to asymmetric designs. If in a symmetric design all interactions and main effects among factors A_1, A_2, ... A_k can be estimated orthogonally within the fraction then deletion of levels of one or more factors does not affect the orthogonality of main effects; if only the k factor interaction is aliased with the fraction then levels may be deleted from up to $k-3$ factors without disturbing the orthogonality of the main effects.

Example 4.11 A 3^{4-1} design yields an orthogonal main effects $2^1 3^3$ design in 18 units but deletion of levels from any other factor causes main effects to be non-orthogonal.

v) Collapsing factor levels. This operation does not disturb orthogonality of main effects but results in unequal replication. It can be used where deletion of levels can not.

Example 4.12 In a 3^{3-1} design with ABC as defining contrast, collapsing level 2 for factors B and C onto level 1 gives the orthogonal main effects design:

0 0 0	1 0 1	2 0 1
0 1 1	1 1 0	2 1 0
0 1 1	1 1 1	2 1 1

The design obtained by deleting level 2 of factors B and C does not have orthogonal main effects.

Extensive tables of orthogonal main effect designs are given by Addelman and Kempthorne (1961) who use some of the above techniques. Their tables are easy to store but also straightforward to generate.

Example 4.13 'PLAN 5' for 25 units (Addelman, 1962) requires:

1) a method for generating an orthogonal main effect design for six five-level factors in 25 units;

2) methods of collapsing factors.

The first stage is achieved simply through the design key matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

and the second stage is readily achieved through an appropriate collapsing routine.

Margolin (1969b) presents constructions for RQ^m designs with orthogonal main effects and resolution V; they are minor variations of the procedures used in Examples 4.8 and 4.10.

Summary. All the procedures described above for obtaining main effect designs are readily included in a small computer program. The orthogonal arrays of Addelman and Kempthorne (1961) for s level

factors in $2s^2$ units are more readily stored than generated.

4.5.3 Response surface designs

Response surface designs form a large class and, by and large, are not readily constructed from simple generators. The experimenter has competing requirements among which are the following:

1) the parameter estimates should be as near orthogonal as possible;

2) the biases of the estimates should be minimized.

3) the variances of the estimates should be minimized.

4) the variance of the predicted y should follow some predetermined pattern, (e.g. flat over a large area near the centre of the design, or equal for all points equidistant from the design centre);

5) the estimates of the position $(x_1, x_2, \ldots, x_k)$ with maximum yield (in a second-order design) need to have small variance.

For larger designs some form of blocking which is (nearly) orthogonal to the model parameters is also required. The resolution of these problems can result in a development of a large specific program, discussion of which is outside the scope of this thesis. However, designs closely related to response surface designs are discussed in this thesis (e.g. §4.4). Construction of some general purpose response surface designs can therefore be achieved through small adjustment to a program of the type we are considering. In Appendix A4.2 we present one suggestion, namely, a procedure, based on selecting defining contrasts (§3.5), is given for constructing central composite designs.

Appendix A4.1

Construction of Hadamard matrices of small order

In this section we note methods of constructing small Hadamard matrices of order $N < 100$ and make a suggestion as to how they should be constructed in a small program.

a) $N = 2^k$: matrix of effects for 2^k factorial (special case of c)

b) $N = p^\alpha + 1$ where p prime: theorem 17.4.3 Raghavarao (1971)

c) $N = 8k$: theorem 17.4.2 Raghavarao (1971)

d) $N = 2(p^\alpha + 1)$, $p^\alpha + 1 = 2 \bmod 4$: theorem 3.3, Hadayat and Wallis (1978)

e) $N \neq 92$: Plackett and Burman (1946)

f) $N = 4(2m+1)$: Williamson construction, Hedayat and Wallis (1978).

Methods a) to f) are applicable throughout the range. But the following strategy suffices for $N \leq 100$:

1) $N = 2^k$: use method a)

2) $N = 4(2m+1)$: use method f)

3) $N = 2^k.4(2m+1)$: use method c) (i.e. direct product).

Appendix A4.2

Construction of central composite second-order designs

In this appendix an outline is given of a simple procedure for constructing a central composite second-order response surface design. It is based on the method of selecting defining contrasts (§3.5). For brevity, we assume that the principal concern of the experimenter is for orthogonality of parameter estimates.

The user is required to specify:

1) the response surface model (including the number of factors, n);

2) the minimum m_o and maximum m_1 number of experimental units.

The procedure then follows the steps:

a) it constructs a basic 2^{n-m} orthogonal factorial design with blocks of the smallest acceptable size 2^{n-m-p} ;

b) determines the number of axial points and their distances from the centre;

c) determines the number of centre points for the design to be as nearly rotatable as possible;

d) allocates units to blocks such that blocking is (almost) orthogonal.

Two features of the model are important during construction. Firstly, the $k \leq n$ quadratic terms indicate factors for which axial points are required. Secondly, the cross-product terms determine the two-factor interactions to be estimated. By specifying effects to be estimated and those already known (i.e. determined elsewhere or assumed negligible) the method of selecting defining contrasts (§3.5) can be used to construct a suitable fraction with smallest block-size.

The final design contains $m_o \leq N = n_b + n_a + n_o \leq m_1$ units where n_b is the number of units in the basic factorial $=2^{n-m}$; n_a is the

number of axial points $=2k$, and n_o is the number of centre points. For orthogonality, the distance α of the axial points from the origin is given by the condition (John (1971, p.204)):

$$\alpha^2 = ((n_b^2 + n_b(n_o + n_a))^{\frac{1}{2}} - n_b)/2 = n_b((N/n_b)^{\frac{1}{2}} - 1)/2 \qquad 1)$$

Let $n' = 4(\sqrt{n_b} + 1) - n_a$ then for rotatability, the distance α satisfies $\alpha^2 = \sqrt{n_c}$ and $n_o = n'$. Select α so that the design is orthogonal and n_o as close to n' as possible (hence α^2 is as close to $\sqrt{n_c}$ as possible for from 1) α^2 increases monotonically with n_o).

There are $2^p + 1$ blocks in all, 2^p for the factorial and one for the axial points. To maintain orthogonality of blocking allocate n_{oc} centre points to each factorial block (John (1971, p.207)):

$$n_{oc} = \frac{n_b(n_a + n_o - 2\alpha^2)}{2\alpha^2 + n_o} \qquad 2)$$

where n_b = the number of units in a block, 2^{n-m-p}. This simplifies to :

$$n_{oc} = 2^{1-p}\alpha^2 \quad . \qquad 3)$$

If n_{oc} is not integer then n_{oc} may be set to the nearest integer and α chosen to satisfy equation 1) or 2) as preferred.

Example: It is required to estimate the full second-order model for $n=6$ factors with 53 or 54 runs.

Step a): The basic factorial is a 2^{6-1} in two blocks of 16 units with defining contrast ABCDEF and confounded effects ABC and DEF.

Steps b) and c): All quadratic terms are required hence $k=6$ and $n_a=12$ and $n_o=9$ or 10. Now $n' = 14.63$ so choose $n_o=10$ and hence $\alpha^2 = 4.875$.

Step d): There are two blocks and so $p=1$ and from 3) $n_{oc} = \alpha^2$. Choosing $n_{oc} = 5$ the resulting design has five centre points allocated to each block and none to the axial block.

CHAPTER 5

CONSTRUCTING LATIN SQUARES

5.1 Summary

Orthogonal Latin squares are used in constructing many other designs including lattice designs (§2.6). In this chapter we present three methods for generating orthogonal Latin squares, the design key method, a cyclic method and a modified cyclic method.

Basic definitions are given in §5.2. In §5.3 methods are given for constructing a single Latin square.

The design key method is described in §5.4 and it is shown to be useful for constructing orthogonal Latin squares for all orders $n \neq 4m+2$. Cyclic Latin squares are shown to provide alternative orthogonal squares for all odd $n < 30$ except $n = 3$, or 9 and more orthogonal squares for $n = 15, 21$. A modified cyclic construction is shown to yield orthogonal Latin squares of order $n = 4m+2$. The cyclic and modified cyclic designs are orthogonal to their conjugates. Tables of generators are supplied in the appendices.

5.2 Definitions

The following notation and definitions are used within this chapter. A Latin square of side or order n is denoted L and the symbol allocated to cell i,j is denoted L_{ij} where $0 \leq i,j,L_{ij} \leq n-1$. The kth diagonal of a Latin square is a set of n cells $(i,j), (i+1,j+1) \ldots (i+n-1,j+n-1)$ where each element is reduced modulo n, and $j-i = k$. The zeroth diagonal is called the leading diagonal and diagonals running from top right to bottom left reverse diagonals. A transversal is a set of cells, one in each row and column such that no two cells contain the same symbol; a Latin square

may have $0 \leq k \leq n$ disjoint transversals.

A diagonal Latin square is one with each diagonal containing a single symbol. In a cyclic Latin square the diagonals are generated cyclically so that $L_{i+1,j+1} = L_{ij} + 1$ modulo n. A modified cyclic square has two sets of symbols $\{T\}$ and $\{X\}$ of order m and $n-m$, $(m \geq n/2)$, and is partitioned

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

where A has dimension m. A has $2m-n$ cyclically generated diagonals with symbols $\{T\}$ and $n-m$ constant diagonals with symbols $\{X\}$; B and C have their rows and columns respectively, cyclically generated with symbols $\{T\}$; D is a Latin square with symbols $\{X\}$ e.g. $n = 6$, $m = 5$, $T = \{0,1,2,3,4\}$, $X = \{I\}$

0	2	I	1	3	4
4	1	3	I	2	0
3	0	2	4	I	1
I	4	1	3	0	2
1	I	0	2	4	3
2	3	4	0	1	I

In a column complete Latin square each symbol has each other symbol occurring exactly once in the square below it. A complete Latin square is both column complete and row complete.

Two Latin squares belong to the same transformation set if one can be derived from the other by permutation of rows, columns and symbols. Two squares are said to be conjugate if the row and column categories are inter-changed, and adjugate if the symbols are inter-changed with either the row or column category. A square which is the same as its conjugate is symmetric.

Two Latin squares are called orthogonal if when one is superimposed

on the other no ordered pair of symbols occurs in more than one cell. A set of mutually orthogonal Latin squares (MOLS) is one in which all pairs of squares are orthogonal. No set of MOLS can contain more than $n-1$ squares and a set of maximum size is called a complete set of MOLS. A Latin square and its conjugate which are orthogonal are said to be a pair of conjugate orthogonal Latin squares (COLS) and each square is called a twin. Twins share a common transversal, namely, the leading diagonal.

5.3 Constructing a single Latin square

5.3.1 Selecting a Latin square at random

For experiments based on Latin squares, Fisher and Yates (1963, Table XV) recommend that the squares be selected at random from a sufficiently large set. They describe a method (Example 1.6, §1.2) based on randomizing rows, columns and treatments of representative squares. For $n \leq 6$ they give tables of representative squares for most transformation sets; those for the remaining sets are conjugates of these. For $n \geq 7$ a single representative is given but others may be obtain by forming adjugate squares.

The randomization, conjugation and adjugation operations are features to be regarded as standard in an experimental plan program (see Chapter 7) and the selection at random of a Latin square of order $n \leq 12$ can therefore be achieved by a small program accessing a file containing Table XV of Fisher and Yates (1963).

5.3.2 Some methods for constructing Latin squares

Simple methods for generating Latin squares are described. Some methods are extended to MOLS in §§5.4 and 5.5.

Example 5.1 The following Latin squares are constructed by methods A, B and C respectively:

0	1	2	3	4	5
1	2	3	4	5	0
5	0	1	2	3	4
2	3	4	5	0	1
4	5	0	1	2	3
3	4	5	0	1	2

(a)

0	1	2	3	4
4	0	1	2	3
3	4	0	1	2
2	3	4	0	1
1	2	3	4	0

(b)

0	3	1	4	2
3	1	4	2	0
1	4	2	0	3
4	2	0	3	1
2	0	3	1	4

(c)

Method A Form the first column then form subsequent columns by adding 1 modulo n to the symbols of the previous column.
This method is useful for constructing column complete Latin squares.

Method B Form a diagonal Latin square with symbol k in the kth diagonal.

Method C (Cyclic Latin square, $n = 2m - 1$). Choose a suitable first row then form subsequent rows as follows:

$$L_{ij} = L_{i-1,j-1} + 1 \pmod{n} \quad i,j = 1,2,\ldots n-1$$
$$L_{io} = L_{i-1,n-1} + 1 \pmod{n} \qquad (1)$$

This construction is always available for $n = 2m - 1$ given suitable choice of the first row (Dénes and Keedwell (1974), p 310). Each diagonal is a transversal and the square is orthogonal to a diagonal square.

Example 5.2 The following squares are constructed by methods D, E and F respectively.

0	I	1	2
2	1	I	0
I	0	2	1
1	2	0	I

(d)

0	α^3	α	α^2
α^2	α	α^3	0
α^3	0	α^2	α
α	α^2	0	α^3

(e)

(00)	(11)	(01)	(10)
(10)	(01)	(11)	(00)
(11)	(00)	(10)	(01)
(01)	(10)	(00)	(11)

(f)

Method D (Modified cyclic square, $n = 2m$). Construct a square of order $n-1$ by method C and replace any diagonal by invariant symbol I (or $n-1$). Border the matrix by an extra row and column placing the symbol removed from the i,j^{th} cell in the i^{th} position of the extra column and the j^{th} position of the extra row. Put symbol I in the remaining cell.

The square has at least one transversal formed from $L_{n-1,n-1}$ plus any diagonal of the sub square other than that containing only symbol I. (The method can be modified to allow replacement of p disjoint transversals from a basic square of order $n-p$ where $2p \leq n$.)

A set S is called a quasigroup if there is a binary operation defined in S and if $a,b,\varepsilon S$ then the equations $a.x=b$ and $y.a=b$ have exactly one solution. The multiplication table of a quasigroup is a Latin square. (Dénes and Keedwell, 1974, p. 16). A group is also a quasigroup but its multiplication table satisfies a quadrangle criterion:

if $a_{jk} = a_{j'k'}$, $a_{ik} = a_{i'k'}$, $a_{i\ell} = a_{i'\ell'}$ then $a_{j\ell} = a_{j'\ell'}$ 2)

(Dénes and Keedwell, 1974, p. 18). These properties extend to finite fields which play a central role in the construction of orthogonal Latin squares.

Method E Let S be a finite group closed under addition with elements $\alpha_0,\alpha_1,\ldots\alpha_{n-1}$ where α_0 is the zero element. Let f_1 and f_2 be one-one functions on S then form L as follows:

$$L_{ij} = f_1(\alpha_i) + f_2(\alpha_j) \qquad 0 \leq i, j \leq n-1 \qquad 3)$$

In particular, if $n = p^k$ for p prime and S is a finite field then

$$L_{ij} = \alpha_1\alpha_i + \alpha_2\alpha_j \qquad \alpha_1,\alpha_2 \neq \alpha_0 \qquad 4)$$

suffices. If n is prime then method E includes methods A, B and C:

method A, set $\alpha_2 = e$, the identity element

B, set $\alpha_2 = -\alpha_1$

C, set $\alpha_2 = e - \alpha_1$

Methods D and E sometimes overlap as in Example 5.2 above.

Method F ($n = p^k$ for p prime). Replace the row, column and treatment factors of the square by k pseudo-factors each with p levels. Denote a $2k \times k$ design key matrix K (§3.3) by

$$K = \begin{bmatrix} A \\ B \end{bmatrix} \qquad 5)$$

where the first k rows correspond to the row pseudo-factors, the last k to column pseudo-factors, and the k columns to treatment pseudo-factors. The design key matrix K will generate a Latin square if A and B have full rank. The proof is trivial; we simply note that the p^k distinct vectors of length k with elements $0,1, \ldots, p-1$ form the elements α of an additive group and multiplication by a non-singular matrix is a one-one function.

Example 5.3 The Latin square in Example 5.2(f) is constructed from the design key matrix $K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}^T$.

Method G (Extending methods E and F when n is not prime power).

When n is not a prime power it can be expressed

$n = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$ where p_i are distinct primes. To extend method E define a system of n elements

$$\alpha = (g_1, g_2 \ldots g_k)$$

where $g_i \varepsilon GF(p_i^{n_i})$ with addition and multiplication

$$\alpha + \alpha' = (g_1, g_2, \ldots, g_k) + (g_1', g_2', \ldots, g_k')$$

$$\alpha\, \alpha' = (g_1 g_1', \ldots, g_k g_k') \qquad 6)$$

If α and α' have no zero coordinates then a square of order n constructed as in 4) above is a Latin square (Raghavarao (1971, p34)).

To extend method F replace the row, column and treatment factors by $S = \sum_k n_i$ prime-level pseudo-factors. Form the design key matrix

$$\begin{bmatrix} A_1 & O & \dots & O \\ O & A_2 & \dots & O \\ \cdot & \cdot & \dots & \cdot \\ O & O & \dots & A_k \\ B_1 & O & \dots & O \\ O & B_2 & \dots & O \\ \cdot & \cdot & \dots & \cdot \\ O & O & \dots & B_k \end{bmatrix}$$

where A_i and B_i are $n_i \times n_i$ non-singular matrices corresponding to pseudo-factors with p_i levels, then this key matrix generates a Latin square.

Notes: i) Latin squares generated by method F satisfy the quadrangle criterion 2) above and the squares generated by different choice of matrices may be derived from each other by permutation of rows and columns; matrices A and B determine the row and column permutations respectively. Elements α_1 and α_2 act similarly with method E.

ii) If n is not prime then not all design key matrices generate Latin squares but any key matrix is permissible which upon reordering the k sets of pseudo-factors can be expressed in the form

$$K = \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1k} \\ O & A_{22} & \dots & A_{2k} \\ \cdot & \cdot & \dots & \cdot \\ O & \cdot & \dots & A_{kk} \\ B_{11} & B_{12} & \dots & B_{1k} \\ O & B_{22} & \dots & B_{2k} \\ \cdot & \cdot & \dots & \cdot \\ O & \cdot & \dots & B_{kk} \end{bmatrix} \qquad 7)$$

where A_{ii} and B_{ii} have full rank. To prove this assume that a symbol occurs twice in one row, then there exist distinct vectors of column pseudo-factor levels $\underline{u}$ and $\underline{w}$ for which $B^T\underline{u} = B^T\underline{w}$, i.e. $B^T\underline{v} = 0$ where $\underline{v} = \underline{u} - \underline{w}$. Write $\underline{v} = (\underline{v}_1, \underline{v}_2 \ldots, \underline{v}_k)$; then $B_{11}^T\underline{v}_1 = 0$ so $\underline{v}_1 = 0$ but then $B_{22}^T\underline{v}_2 = 0$ etc. Continuing this process gives $\underline{v} = \underline{0}$ but this is a contradiction so no row contains the same symbol twice. A similar result can be applied to columns to complete the proof.

5.4 Orthogonal Latin squares

Orthogonal Latin squares are used to constructa variety of other designs including BIB, PBIB, generalized lattice and change-over designs. Some of these relationships are mentioned elsewhere in this thesis but others are given by Hedayat and Shrikande (1971), Raghavarao (1971), Vajda (1967) and Dénes and Keedwell (1974).

In §§5.4 and 5.5 we discuss methods for constructing mutually orthogonal Latin squares (MOLS) of order $n \leq 30$. We are not concerned with theorems about the maximum number of MOLS, but concentrate on a few simple methods which generate a wide range of MOLS. We find that two methods can between them construct the maximum known number of MOLS (N(n)) for all $n \leq 30$ except $n = 12$ and 24 (Appendix A5.1). The design key method, which is useful for all values of n except $n = 4m + 2$, is discussed in this section. The cyclic/modified cyclic method which is useful for $n = 4m + 2$ is discussed in §5.5.

5.4.1 Constructing complete sets of MOLS

For $n = p^k$ the following procedure may be used to construct a complete set of MOLS by the design key method. Replace the row,

column and treatment factors by k pseudo-factors with p levels then Latin squares L_1 and L_2 produced by design key matrices K_1 and K_2 are orthogonal if $(K_1\ K_2)$ has full rank. In particular, if

$$K_1 = \begin{bmatrix} A_1 \\ I \end{bmatrix} \qquad K_2 = \begin{bmatrix} A_2 \\ I \end{bmatrix}$$

then L_1 and L_2 are orthogonal when $A_1 - A_2$ has full rank. A set of matrices $\{A_i\}$ which form a finite field can be used to construct a complete set of MOLS for apart from the matrix $A_0 = 0$ the remaining matrices have full rank as does any matrix $A_i - A_j$. Such a field exists for all p^k and the finite field and design key methods are equivalent (see Appendix A5.2)

Let A be a primitive root of the field then the Latin squares L_i generated by the design key matrices K_i where

$$K_i = \begin{bmatrix} A^i \\ I \end{bmatrix} \qquad i = 1,2, \ldots, n-1$$

form a complete set of MOLS. If n is prime then A^i are residues modulo n. For $n = p^k < 30$, $k \neq 1$ primitive root matrices A are presented in Appendix A5.3 which correspond to the fields reported by Kempthorne (1952).

Example 5.4 A pair of MOLS of order $n = 25$ is required. Matrices

$$A = \begin{bmatrix} 1 & 3 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad A^2 = \begin{bmatrix} 4 & 1 \\ 2 & 4 \end{bmatrix}$$

belong to a field $GF(5^2)$ generated by A and therefore design key matrices

$$\begin{bmatrix} 1 & 3 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 4 & 1 \\ 2 & 4 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

generate a pair of orthogonal Latin squares.

For non-prime values of n, Latin squares L_r constructed by method G (§5.3.2) such that the (i,j)th element of L_r is

$$L^r_{ij} = \alpha_r \alpha_i + \alpha_j$$

are mutually orthogonal if for each pair $r = s$, t, $\alpha_s - \alpha_t$ contains no zero elements (Raghavarao (1971, p 34)). The equivalent condition for design key matrices is that Latin squares constructed from matrices

$$K_r = \begin{bmatrix} A_{1r} & O & \dots & O \\ O & A_{2r} & \dots & O \\ . & . & \dots & . \\ O & . & \dots & A_{kr} \\ I & . & \dots & O \\ . & . & \dots & . \\ O & . & \dots & I \end{bmatrix}$$

are mutually orthogonal if $A_{ir} - A_{is}$ are non-singular for all i. We call this the MacNeish-Mann construction. The maximum number of orthogonal squares which can be generated is determined by the smallest number for any $p_i^{n_i}$; the method is therefore not available for any order $n = 4m + 2$.

5.5 A cyclic generator for mutually orthogonal and conjugate orthogonal Latin squares

In this section we consider cyclic Latin squares (method C, §5.3.) which do not necessarily satisfy the quadrangle criterion and cannot therefore be generated from design key matrices.

The first row of a cyclic Latin square is also the first column of its conjugate and hence a single row suffices to generate the two squares. A pair of cyclic COLS are both orthogonal to a diagonal square and a single row then suffices to generate a set of three MOLS.

Cyclic Latin squares cannot be generated for even values of n but can be for all odd values. Cyclic COLS have been constructed for all odd $n < 30$ except $n = 3$, and 9. (It is known that none exist for these values). All cyclic COLS for $n = 5$ and 7 may be generated by the design key method. For all other $n < 30$ the first rows of representative COLS have been given in Appendix A5.6. The construction for three orthogonal squares of order 15 and four of order 21 (Appendix A5.5) match the maximum number of orthogonal squares yet constructed for these orders (denoted $N(n)$).

A method for determining suitable first rows is given in §5.5.2

5.5.1 A modified cyclic generator for mutually orthogonal and conjugate orthogonal Latin squares

In this section we extend the modified cyclic method (method D, §5.3) to orthogonal Latin squares of order $2m$. The squares generated do not necessarily satisfy the quadrangle criterion. Of particular interest are squares of order $4m + 2$.

The rationale of the method may be demonstrated by an example:

Example 5.5: Consider the three mutually orthogonal squares for $n=4$.

0	2	3	1	0	3	1	2	0	1	2	3
3	1	0	2	2	1	3	0	1	0	3	2
1	3	2	0	3	0	2	1	2	3	0	1
2	0	1	3	1	2	0	3	3	2	1	0

The squares can be partitioned as shown; they can be seen to contain

a) a cyclic Latin square of order $n = 3$ with symbols 1,2,3 but with one diagonal replaced by 0

b) a top and left border containing the missing diagonal

c) the element 0 in the first cell.

The second square is the conjugate of the first; the last is symmetric.

The process can be reversed and starting from a cyclic Latin square of order 3 a set of 'modified' cyclic squares of order $n = 4$ which are mutually orthogonal can be constructed. Starting with squares of order 7 a complete set of orthogonal squares of order $n = 8$ can be formed (Appendix A5.6). As with cyclic squares each pair of modified cyclic COLS can be generated from a single row.

The modified cyclic method is related to the method used by Bose, Shrikande and Parker (1960) to construct the first known pair of MOLS of order 10; it is also related to the sum-composition method of Hedayat and Seiden (1974). The current requirement for a simple general method for constructing COLS of all orders $4m + 2 \leq 30$ differs from the requirements of these authors and a different approach is adopted. To highlight the similarity between the three approaches the borders are moved to the right and bottom, the symbols are renumbered and the extra symbol is denoted I or x (see Example 5.2 (d)).

More than one diagonal in the basic cyclic square can be replaced. Thus if $n = p+q$ where p is odd, $p > 2q$ and there exist orthogonal Latin squares of order q the construction can be based on replacing q diagonals from a cyclic square of order p. For brevity, in the current discussion attention is restricted to $q = 1$ but we note that when $n \geq 20$ and $q \geq 5$ the pair of COLS can sometimes have at least five common transversals.

5.5.2 Constructing the first row of orthogonal cyclic and modified cyclic Latin squares

a) Cyclic Latin squares

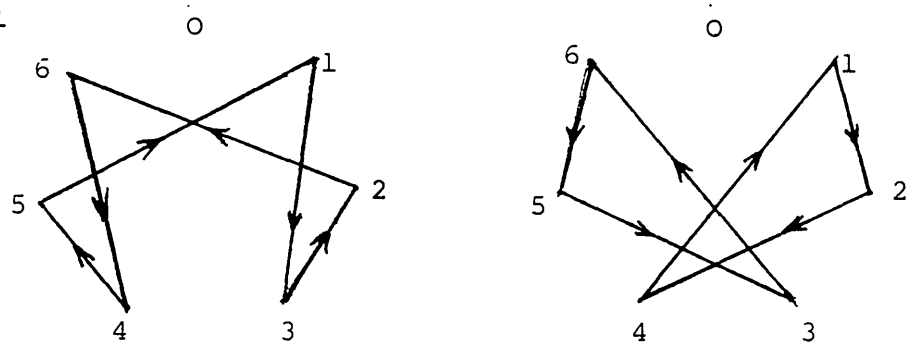
In this section we give a brief outline of the construction leaving details to Appendix A5.2. Let A and B be two cyclic

Latin squares of order $n = 2m-1$ with first rows $a_o, a_1 \ldots a_{n-1}$ and $b_o, b_1 \ldots b_{n-1}$ respectively; without loss of generality a_o and b_o can be set to zero. To satisfy the condition of a Latin square the a_i must satisfy the relation

$$a_{i+j} - a_i \neq j \quad \text{i.e.} \quad a_i - i \neq a_j - j \quad \text{if} \quad i \neq j \qquad 1)$$

and similarly the b_i. For each diagonal $a_i - b_i = d_i$ is constant; the squares A and B are orthogonal if and only if no two d_i are equal. The search for a set $\{a_i, b_i\}$ may be represented graphically by a set of n points upon which n-1 lines are superimposed. Each line connects two points and each point lies on two lines (except one point denoted zero which has no line through it). The lines are measured directionally and no two have the same length.

Example 5.6



The first graph defines a circuit $1-3-2-6-4-5-1$ and the pairings (1,3), (3,2), (2,6), (6,4), (4,5), (5,1). If these pairs are ordered so as to satisfy the conditions for both to be Latin squares then the resulting squares are orthogonal.

For A and B to form a pair of COLS there is an extra condition

$$a_i - b_{n-i} = i$$

$$\text{or} \quad a_i - b_i = b_{n-i} - a_{n-i} \qquad 2)$$

The following simple procedure checks whether this condition is satisfied. Ignoring the zero difference the remaining differences $d_i = 1, 2 \ldots n-1$ can be paired d and d' such that $d + d' = n$.

Set

$$a - b = d$$

$$a' - b' = d'$$

and put $e = a - b'$, $e' = a' - b = n - e$. The condition is satisfied if all non-zero values for e or e' occur exactly once. For any pair $a - b' = e$, a is placed in location e and b' in location $n - e$.

Example 5.7 For the first graph above the six differences d_i are

$$d_1 = (3,2) \quad d_6 = (4,5)$$

$$d_2 = (6,4) \quad d_5 = (1,3)$$

$$d_3 = (2,6) \quad d_4 = (5,1)$$

The corresponding e,e' are $5,2$; $3,4$; $1,6$ respectively and so a cyclic COLS can be derived. The first rows of A and B are

A	0	2	4	6	1	3	5
B	0	6	5	4	3	2	1

(The squares correspond to the squares formed by design key matrices $\binom{6}{2}$ and $\binom{2}{6}$ respectively.)

b) Modified cyclic Latin squares

For orthogonal modified cyclic Latin squares of order $n = 2m$ with a single border row and column, one element of the first row of each square is moved to the border and replaced by symbol x. The elements a_i and b_j, say, which are moved cannot come from the same positions (i.e. $i \neq j$) and consequently two differences d_i and d_j, say, corresponding to the affected diagonals, are formed by the border row and column. For twins (§5.2) the replacement of element a_i implies the replacement of element b_{n-i}.

The construction procedure follows very closely that outlined

above for cyclic squares of order 2m-1 but not all non-zero pairings e,e' are wanted. If m-2 pairs of e,e' and one pair d,d' can be chosen so that all non-zero differences are represented then a pair of COLS can be constructed, the cells containing the x's being determined by the d,d' chosen. (More generally, if there exists an orthogonal square (twin) of order q and if q pairs d,d' and m-q-1 pairs e,e' can be chosen then orthogonal squares (twins) of order 2 m+q+1 can be constructed.)

Example 5.8: For n=9 the path 1-2-5-3-8-7-4-6-1 contains all non-zero differences

$d_1 = (8,7)$ $d_8 = (1,2)$ e,e' = 6,3

$d_2 = (5,3)$ $d_7 = (4,6)$ e,e' = 8,1

$d_3 = (7,4)$ $d_6 = (2,5)$ e,e' = 2,7

$d_4 = (3,8)$ $d_5 = (6,1)$ e,e' = 2,7 .

The e values from rows 1, 2 and 3 and the d values from row 4 give all non-zero differences. Either d_4 or d_5 may be split to give two pairs of COLS of order 10. These have first rows

	A_1	0	4	7	1	3	x	8	2	5	6
$(A_1^T =)$	B_1	0	6	4	2	x	8	7	5	3	1

	A_2	0	4	7	1	x	6	8	2	5	3
$(A_2^T =)$	B_2	0	6	4	2	1	x	7	5	3	8

and the first pair yield COLS

0	4	7	1	3	x	8	2	5	6
6	1	5	8	2	4	x	0	3	7
4	7	2	6	0	3	5	x	1	8
2	5	8	3	7	1	4	6	x	0
x	3	6	a	4	8	2	5	7	1
8	x	4	7	1	5	a	3	6	2
7	0	x	5	8	2	6	1	4	3
5	8	1	x	6	0	3	7	2	4
3	6	0	2	x	7	1	4	8	5
1	2	3	4	5	6	7	8	0	x

0	6	4	2	x	8	7	5	3	1
4	1	7	5	3	x	0	8	6	2
7	5	2	8	6	4	x	1	0	3
1	8	6	3	0	7	5	x	2	4
3	2	0	7	4	1	8	6	x	5
x	4	3	1	8	5	2	0	7	6
8	x	5	4	2	0	6	3	1	7
2	0	x	6	5	3	1	7	4	8
5	3	1	x	7	6	4	2	8	0
6	7	8	0	1	2	3	4	5	x

Appendix A5.1

Numbers of mutually orthogonal Latin squares ($3 \leq n \leq 30$)

n	p^k	N(n)*	Design Key	Cyclic/modified cyclic
3	yes	2	2	1
4	yes	3	3	3
5	yes	4	4	4
6	no	1	1	1
7	yes	6	6	6
8	yes	7	7	7
9	yes	8	8	1
10	no	2	1	2
11	yes	10	10	10
12*	no	5	2	2
13	yes	12	12	12
14	no	2	2	2
15	no	3	2	3
16	yes	15	15	2
17	yes	16	16	16
18	no	2	1	2
19	yes	18	18	18
20	no	3	3	2
21	no	4	2	4
22	no	2	1	2
23	yes	22	22	22
24*	no	3	2	2
25	yes	24	24	2
26	no	2	1	2
27	yes	26	26	0
28	no	3	3	2
29	yes	28	28	28
30	no	2	1	2

*For $n = 12, 24$ neither design key nor modified cyclic method achieve N(n), the maximum number of MOLS so far obtained.

Appendix A5.2

The equivalence of the finite field and design key methods

Two theorems are presented which establish the equivalence of the finite field method and the design key method for generating orthogonal Latin squares of order $n=p^k$.

Theorem A5.2.1 Let S be a group with elements $\{\alpha_0, \ldots, \alpha_{n-1}\}$ and operation + and $H=\{h_1,h_2, \ldots, h_k\}$ denote a set of automorphisms on S. Form two squares L_1 and L_2 with $(i,j)^{th}$ cell

$$L_1 : \quad h_2(\alpha_i)+\alpha_j$$

$$L_2 : \quad h_2(\alpha_i)+\alpha_j \quad h_1 \neq h_2;\ i,j\ 0,1, \ldots, n-1$$

Define the sum and product of h_r and h_s to be

$$h_r + h_s(\alpha) = h_r(\alpha) + h_s(\alpha)$$

$$h_r o h_s(\alpha) = h_r(h_s(\alpha))$$

If $G=H \cup \{\emptyset\}$ where $\emptyset(\alpha)=0$ all α, S is a finite field then L_1 and L_2 are orthogonal Latin squares.

Proof It is shown in §5.3.1 that L_1 and L_2 are Latin squares. They are orthogonal if and only if each pair of treatments x_1 and x_2 occur in exactly one cell i.e. if the equations

$$x_1 = h_1(\alpha_i) + \alpha_j$$

$$x_2 = h_2(\alpha_i) + \alpha_j$$

have exactly one solution for (α_i,α_j). The solution to these equations is

$$\alpha_i = h'(x_1-x_2)$$

$$\alpha_j = h''(h_1^{-1}(x_1) - h_2^{-1}(x_2))$$

where $h' = (h_1-h_2)^{-1}$ and $h'' = (h_1^{-1}-h_2^{-1})^{-1} \in H$. The existence of h_1^{-1}, h_2^{-1}, h' and h'' follow because G is a finite field. The solution is unique because of the automorphism properties of the h_r ; the treatment pair x_1,x_2 occur in only one cell.

Example: Let $h_r(\alpha_i) = \alpha_r \alpha_i$, $\alpha_r \neq 0$ then $G = H \cup \{\emptyset\}$ is GF(n). This is the finite field construction for MOLS (Raghavarao (1971, p.3)).

Theorem A5.2.2 If $n = p^k$ then there exists a complete set of MOLS generated by design key matrices $[A^i, I]^T$ for some $k \times k$ matrix A.

Proof Let h_r be a non-singular $k \times k$ matrix with elements modulo p and let α_i be denoted by a $k \times 1$ vector with elements modulo p. Define $h_r(\alpha_i)$ by the product of h_r and α_i (with reduction modulo p) then h_r is an automorphism on S.

A field $G = GF(p^k)$ of non-singular matrices h_r can be constructed as follows. Each element in a known field G_o can be written in the form

$$a_i = a_{io} + a_{il}x \ldots + a_{i(k-1)}x^{k-1}$$

with a_i being residues modulo p. Addition is defined in the obvious way and multiplication is defined by the product of two polynomials followed by reduction modulo a selected irreducible polynomial $q(x) = \sum^{k} b_j x^j$, $(b_k \neq 0)$. Let $\tilde{a}$ be a primitive root of the field and form a $k \times k$ matrix A whose ith row contains the coefficients a_{ij} of $\tilde{a}_i = \tilde{a}x^i$ then A is also a primitive root of a field G with $h_r = A^r$ (D. Monk, private communication 1981). G and G_o are equivalent due to the uniqueness of $GF(p^k)$.

Example: A field G_o for $p = 2$, $k = 4$ can be constructed from an irreducible polynomial $1 + x + x^4$ and primitive root x. The equivalent matrix field is generated cyclically from primitive root

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Appendix A5.3

Design key matrices for mutually orthogonal Latin squares ($3 \leq n < 30$)

1) n = prime

1a) Row permutation:

The $n-1$ design key matrices $\begin{bmatrix} i \\ 1 \end{bmatrix}$, $i = 1, 2, \ldots, n-1$ generate a complete set of MOLS, by permuting rows other than the first. For $i^2 \neq 0, 1$ the squares are orthogonal to their conjugates.

1b) Cyclic method:

The $n-2$ design key matrices $\binom{i}{1-i}$, $i \neq 0,1$ generate a set of cyclic MOLS. For $i \neq \frac{n+1}{2}$ the squares generated by design key matrices $\binom{i}{1-i}$ and $\binom{1-i}{i}$ form a pair of COLS. The square formed from the design key matrix $\binom{1}{n-1}$ completes the set.

2) n = prime power = p^k

A field of p^k elements can be constructed from $k \times k$ matrices with elements modulo p. Let A be the 'primitive' matrix from which the multiplicative group of the field is generated, then methods 1a) and 1b) carry through by replacing 1 by I_k, $n-1$ by $-I_k$ and i by A^i. The cyclic properties of method 1b) no longer hold. Conditions for COLS are changed slightly: they become $A^{2i} \neq 0$ or I and $A^i \neq \frac{n+1}{2} I$ for methods 1a and 1b respectively. The primitive matrices presented below correspond to the primitive roots given by Kempthorne (1952, p. 334)

Primitive matrices

$$n = 4 \qquad A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$n = 8 \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$n = 9 \quad A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

$$n = 16 \quad A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

$$n = 25 \quad A = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$$

$$n = 27 \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

3) n = 15, 21

The method used is based on the McNeish-Mann approach (§5.4.1) using either method 1a) or 1b) above. The squares generated are not twins. For each square n is factorised into primes and the treatment and block factors replaced by pseudo-factors; the order of the pseudo-factors is noted.

$$n = 15 \quad A_i = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \qquad (3, 5)$$

$$n = 21 \quad A_i = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix} \qquad (3, 7)$$

4) n = 4k

The construction of these squares is based on the McNeish-Mann approach using either method 1a) or 1b). For n = 12, 24 there are two orthogonal squares while for n = 20, 28 there are three orthogonal

squares (of which the first two shown are twins. For each key matrix n is factorised into primes and the treatment and block factors converted to pseudo-factors. The order of the pseudo-factors is noted.

$$n = 12 \quad A_i = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{pseudo-factors } (2, 2, 3)$$

$$n = 20 \quad A_i = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 4 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} (2, 2, 5)$$

$$n = 24 \quad A_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad (2, 2, 2, 3)$$

$$n = 28 \quad A_i = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} (2, 2, 7)$$

5) $n = 4m + 2$

The design key method cannot generate a pair of orthogonal squares.

Appendix A5.4

Construction of cyclic and modified cyclic conjugate orthogonal Latin squares.

a) Cyclic squares

Let A,B be two cyclic Latin squares of order n $(=2m-1)$ with first row $a_o, a_1 \ldots a_{n-1}$ and $b_o, b_1 \ldots b_{n-1}$ respectively $(0 \leq a_i, b_i \leq n-1)$. Without loss of generality we take $a_o = b_o = 0$. Then A will be a Latin square if

$$a_i + j \neq a_{i+j} \quad \text{modulo } n \quad i,j = 0, 1 \ldots n-1$$
$$\text{i.e.} \quad a_i - i \neq a_j - j \quad \text{modulo } n \quad i,j = 0, 1 \ldots n-1 \qquad 1)$$

and similarly for B. The ith diagonals of A and B contain the elements $a_i + j$ and $b_i + j$ $(j = 0, \ldots n-1)$; the two squares are orthogonal if

$$a_i - b_i = a_{i'} - b_{i'} \Rightarrow i = i' \qquad 2)$$

Each difference $d_i = a_i - b_i = 1,2 \ldots n-1$ must therefore occur for one pair a_i, b_i $(i=1,2 \ldots n-1)$. The construction of suitable rows $\underline{a}$ and $\underline{b}$ can be represented by a graph of n points (representing the symbols) are joined by $n-1$ directed lines (representing the differences $a_i - b_i$) each of different non-zero length; each line joins two points and each point other than 0 lies on two lines. Two symbols joined by a line form a couplet $(\bar{a}_s, \bar{b}_s)$ if the couplets can be arranged so that condition 1) is satisfied for both A and B the squares are orthogonal.

If A and B are twins then the first row of B is the first column of A and a further condition holds

$$b_i - a_{n-i} = a_i - b_{n-i} = i \qquad 3)$$

from which it follows

$$b_i - a_i = a_{n-i} - b_{n-i}$$

Given a circuit constructed as above and value d_s , $1 \leq d_s \leq m-1$ there exists a pair of couplets $(\bar{a}_s, \bar{b}_s)$ and $(\bar{a}'_s, \bar{b}'_s)$ satisfying

$$\bar{a}_s - \bar{b}_s = d_s$$
$$\bar{a}'_s - \bar{b}'_s = n - d_s \qquad 4)$$

Form e_s such that

$$\bar{b}_s - \bar{a}'_s = e_s = \bar{a}_s - \bar{b}'_s \ . \qquad 5)$$

If each value $1 \ldots n-1$ belongs to one of the pairs $(e_s, n-e_s)$, $s = 1 \ldots \frac{n-1}{2}$ then the first row of the square A and B are constructed as follows: Let $k = e_s$

$$a_o = 0 \qquad b_o = 0$$
$$a_k = \bar{a}_s \qquad b_k = \bar{b}_s$$
$$a_{n-k} = \bar{a}'_s \qquad b_{n-k} = \bar{b}'_s \qquad 6)$$

It remains to show that both A and B are Latin squares. Note that

$$e_s = \bar{a}_s - \bar{b}'_s$$

it follows

$$\bar{b}'_s = a_k - k \qquad 7)$$

but the $\bar{b}'_s$ are distinct and so from 1) A is a Latin square. The proof for B follows similarly.

Remark

Once a suitable path has been found the following operations yield another:

i) reverse the directions of the arrows

ii) form a mirror image by subtracting each point from n .

iii) multiply each point by a number co-prime to n .

The first operation merely determines which twin is labelled A. The second operation means that reading the first row apart from the zero element of a twin in reverse order and subtracting from n yields another twin (though not necessarily distinct).

b) Modified cyclic squares

Let $n = p+q$ where $p = 2m-1$ then a modified cyclic Latin square which is orthogonal to its conjugate can be obtained by replacing $q \leq m-1$ diagonals of a cyclic square and bordering appropriately (§5.5.1). (We are particularly interested in the case $q = 1$) The construction of suitable first rows for modified cyclic squares is similar to that for cyclic squares (Example 5.8).

Follow the above procedure for cyclic squares as far as stage 5), i.e. form the d_s and e_s $(s = 1,2...m-1)$. For each s choose either d_s or e_s, selecting q d_s values in all, such that each symbol $1,2...n-1 \in \{\pm d_s, \pm e_s\}$ then a pair of orthogonal Latin squares of order $n = p+q$ can be constructed as follows.

For each s where e_s is selected allocate the symbols to the cells a_k, b_k as shown in stage 6) of the cyclic COLS procedure. Let the jth couplet for which d_s is selected be denoted $(\bar{a}_r, \bar{b}_r)$ with difference d_r. Either couplet $(\bar{a}_r, \bar{b}_r)$ or $(\bar{a}'_r, \bar{b}'_r)$ may be selected to generate the jth extra column and the other couplet is then split to couple with symbol x_j in the main subsquare. If $(\bar{a}_r,\bar{b}_r)$ are chosen to generate the extra column then writing $k = d_r$ allocate symbols $(x_j, \bar{b}'_r)$ to cells (a_k,b_k) and symbols $(\bar{a}'_r,x_j)$ to cells (a_{n-k}, b_{n-k}). Alternatively allocate $(\bar{a}'_r,\bar{b}'_r)$ to the jth extra column, $(\bar{a}_r,x_j)$ to cells (a_k,b_k) and $(x_j,\bar{b}_r)$ to cells (a_{n-k},b_{n-k}).

Once the first rows have been formed the sub-square of order p is generated cyclically with invariant diagonals containing the x_j. The extra columns and rows are generated cyclically and a $q \times q$ orthogonal squares containing symbols x_j are added.

If A and B are Latin squares they are orthogonal for all differences e_s occur as diagonals of the $p \times p$ square; all

differences d_s in the row or column borders, all differences between symbols (0, 1, ..., p-1) and symbols $(x_1, x_2, \ldots, x_q)$ occur in the replaced diagonals. The symmetry of construction ensures that if the sub-squares of order q are twins then B is the conjugate of A. It remains to prove A and B are Latin squares. The elements of the first rows are all distinct, and it follows from the construction that A is a Latin square if all $a_k - k$ are distinct. For the a_k derived from the e_s differences the proof follows that given for cyclic squares. For cells derived from d_r differences, $a_k - k = \bar{a}_r - d_r = \bar{b}_r$ which are all distinct. Moreover, the $\bar{b}$ values associated with the d_r are distinct from those associated with the e_s and so A is a Latin square.

Appendix A5.5

Construction of cyclic mutually orthogonal Latin squares for n = 21

There exists a cyclic symmetric BIB (21,5,5,21;1) design generated from the initial block (0, 3, 4, 9, 11). The existence of this design implies the existence of a set of four MOLS of order 21 (Vajda (1967, p 45)). We construct them as cyclic squares.

Consider the five blocks of the BIB design in which symbol 0 occurs

0	3	4	9	11
18	0	1	6	8
17	20	0	5	7
12	15	16	0	2
10	13	14	19	0

1)

Each of these blocks defines a difference set in which all non-zero differences occur. Consider the cyclic Youden square

4	2	3	5
5	3	4	1
1	4	5	2
2	5	1	3
3	1	2	4

2)

and draw a graph connecting the points of the ith block of the BIB design in the sequence determined by the ith row of the Youden square. The resulting graph contains five closed paths and has the properties required to produce cyclic COLS:

9	3	4	11
8	1	6	18
17	5	7	20
15	2	12	16
14	10	13	19

3)

From these paths the initial rows for twins can be constructed.

A 0 18 16 11 9 20 8 17 6 4 19 3 15 14 13 12 2 7 1 10 5

B 0 6 12 4 11 7 18 20 1 3 13 9 16 19 10 2 15 5 8 14 17

Squares A and B are orthogonal to the diagonal matrix with initial row

C 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

and all three orthogonal to the cyclic symmetric square with initial row

D 0 8 15 9 3 17 1 5 18 11 14 4 2 10 19 16 12 20 6 13 7

Appendix A5.6

Initial rows for cyclic and modified cyclic conjugate orthogonal Latin squares

We give initial rows for a cyclic or modified cyclic twin; the first row of the conjugate square can be obtained simply.

Cyclic squares

For n prime an initial row with $a_i = k\,i$, $k \neq 0, 1, (n+1)/2$ $(i = 0, 1 \ldots n-1)$ produces a cyclic twin which satisfies the quadrangle criterion; the squares derived from the initial rows given here do not satisfy the criterion. No attempt is made to present all solutions but each solution gives rise to another (not necessarily distinct) of the form

$$0 \quad n-a_{n-1} \quad n-a_{n-2} \quad \ldots \quad n-a_1 .$$

Each pair of COLS is orthogonal to a diagonal square.

$n = 3, 9$ no solution; $n = 5, 7$ all solutions given by design key method.

$n = 11$

0 7 5 10 6 2 4 1 9 8 3

0 6 9 7 2 8 5 4 3 10 1

$n = 13$

0 9 5 12 6 10 4 8 2 1 7 10 3

0 4 6 9 12 3 8 1 5 10 2 7 11

0 4 6 1 11 10 5 8 3 2 12 7 9

$n = 15$

0 14 13 6 11 7 5 2 12 10 1 4 9 3 8

0 2 13 8 10 12 1 5 7 11 4 14 9 6 3

0 5 8 6 14 4 2 1 13 10 12 3 9 11 7

n = 17

0 11 13 1 8 14 7 2 5 12 15 10 3 9 16 4 6

0 4 12 1 9 14 7 11 2 15 6 10 3 8 16 5 13

0 12 6 8 3 2 13 16 10 7 1 4 15 14 9 11 5

n = 19

0 14 13 1 7 10 2 3 15 8 11 4 16 17 9 12 18 6 5

0 7 5 1 8 13 15 2 9 3 16 10 17 4 6 11 18 14 12

0 8 15 1 5 16 10 12 17 6 13 2 7 9 3 14 18 4 11

n = 21 (see Appendix A5.5).

0 6 12 4 11 7 18 20 1 3 13 9 6 19 10 2 15 5 8 14 17

n = 23

0 7 10 1 17 8 2 12 20 18 9 4 19 14 5 3 11 21 25 6 22 13 16

n = 25

0 8 19 1 16 11 2 33 13 20 7 10 21 4 15 18 5 12 3 23 14 9 24 6 17

n = 27

0 26 4 2 5 22 14 12 19 24 23 21 6 20 18 7 25 10 13 16 11 17 9 8 3 1 15

n = 29

0 21 23 1 20 19 2 26 13 15 22 4 11 24 17 12 5 18 25 7 14 16 3 27 10 9 28 6 8

Modified cyclic squares

The first rows of modified cyclic twins are given for $n = 2m \leq 30$. Construction of the squares from initial rows is described in §§5.5.1 and 5.5.2. Each row represents two alternative first rows -

either choose the first symbol in all brackets or the second. Further rows may be constructed by subtracting each element, other than x, from n - 1 and reversing the order of elements $a_1, a_2, \ldots, a_{n-2}$. For n = 10 six distinct paths have been found, each gives rise to three solutions derived from multiplying the points by 1, 2 and 4, giving the 18 solutions presented.

n = 4

0 (x2) (1x) (21)

n = 6

no solution

n = 8

0 (5x) 4 1 3 6 (x2) (25)

0 6 (3x) 5 1 (x4) 2 (43)

0 4 6 (2x) (x5) 3 1 (52)

Note: if the first symbol in each bracket is taken then the above rows form six MOLS. The set is completed by symmetric Latin square

x 5 3 2 6 1 4 0

n = 10

Set A

a) 0 2 4 (7x) 1 8 (x5) 3 6 (57)

a) 0 7 4 (1x) 8 6 (x5) 3 2 (51)

a) 0 3 5 (1x) 8 6 (x2) 4 7 (21)

b) 0 5 1 (8x) 6 3 (x7) 4 2 (78)

b) 0 6 1 (5x) 2 8 (x7) 4 3 (75)

b) 0 7 3 (5x) 2 8 (x1) 6 4 (15)

Set B

a) 0 (3x) 8 7 5 4 2 1 (x6) (63)

b) 0 (3x) 1 6 8 2 7 5 (x4) (43)

c) 0 (3x) 8 1 7 6 5 2 (x4) (43)

d) 0 (3x) 7 4 1 8 5 2 (x6) (63)

a) 0 8 (6x) 4 7 2 5 (x3) 1 (36)

b) 0 4 (6x) 5 2 1 3 (x8) 7 (86)

c) 0 3 (6x) 1 7 4 2 (x8) 5 (86)

d) 0 7 (6x) 1 5 4 8 (x3) 2 (36)

a) 0 4 7 1 (3x) (x6) 8 2 5 (63)

b) 0 2 8 6 (3x) (x7) 1 5 4 (73)

c) 0 8 6 4 (3x) (x7) 2 1 5 (73)

d) 0 8 5 7 (3x) (x6) 2 4 1 (63)

Notes: i) For squares a) and d) in Set B, the 'reversal' process does not produce new squares.

ii) In each set rows prefixed by the same letter have been derived as multiples of the same underlying path.

n = 12

0 (8x) 10 7 3 6 9 5 2 4 (x1) (1 8)

0 2 (4x) 7 1 10 9 5 3 (x8) 6 (8 4)

0 4 8 7 1 (x3) (5x) 9 2 10 6 (3 5)

n = 14

0 (9x) 5 8 10 2 4 6 12 3 11 7 (x1) (1 9)

0 5 12 (6x) 2 10 1 9 7 3 (x11) 4 8 (11 6)

0 4 6 12 9 (3x) 8 1 (x5) 2 11 10 7 (5 3)

n = 16

0 14 (13x) 6 3 12 11 9 1 10 4 8 7 (x2) 5 (2 13)

0 12 9 (13x) 5 8 4 1 14 11 7 10 (x2) 6 3 (2 13)

0 4 7 1 12 9 (5x) 13 2 (x10) 6 3 14 8 11 (10 5)

n = 18

0 9 4 10 14 (11x) 5 12 2 7 (x6) 13 3 8 1 15 (6 11)

n = 20

0 17 4 8 18 15 10 14 (x7) 3 16 (9x) 3 2 6 5

12 1 11 (79)

n = 22

0 18 4 (12x) 20 15 7 5 13 2 17 19 16 10 8 14 6 9

(x3) 1 11 (3 12)

n = 24

0 10 4 2 19 16 11 (8x) 15 22 20 7

6 21 18 13 (x5) 14 9 12 3 1 17 (5 8)

n = 26

0 17 7 1 11 (20x) 2 6 22 12 16 10 4 21 15 9 13

3 19 23 (x5) 14 24 18 8 (5 20)

n = 28

0 7 11 1 21 13 2 19 22 12 (9x) 4 17 24 3 10

23 (x18) 15 5 8 25 14 6 26 16 20 (18 9)

n = 30

0 16 4 2 5 9 15 14 22 27 26 24 6 23 (25x)

(x13) 21 7 11 10 28 17 19 18 12 8 3 1 20 (13 25)

CHAPTER 6

DESIGNS FOR NON-INDEPENDENT OBSERVATIONS

6.1 Summary

In this chapter we discuss a large class of designs in which observations on any unit may be affected not only by the treatment applied directly but also by treatments applied to 'neighbouring' units where neighbouring may apply to space or to a preceding period. The three types of design considered are i) change-over designs, ii) superimposed designs and iii) serially balanced designs. For each type of design constructions are described and, because they do not fit the standard pattern, analyses outlined.

In §6.2 the designs are introduced and the terminology defined. In §6.3 the relevant literature is reviewed. Methods for constructing change-over designs are discussed in §6.4 with special reference to the designs given by Patterson and Lucas (1962) and the cyclic change-over designs of Davis and Hall (1969). Superimposed designs are discussed briefly in §6.5.

The construction of serially balanced designs is considered in detail (§6.6) because the use of designs of this type in agricultural experiments appears to be increasing. Extensive tables of designs of practical size are given in Appendices A6.4 to A6.6. A description of the analysis of a serially balanced design under a first-order Markov model and with residual effects is given so that aspects of importance in the construction may be highlighted.

6.2 Introduction and definitions

Designs discussed in this chapter are all in use in agricultural

research but have, for the most part, been developed since the publication of the book by Cochran and Cox (1957). Change-over designs are used in animal experiments especially when treatments are expected to have a residual effect. Superimposed designs are used mainly with perennial crops when a second experiment is imposed on the experimental material and residual effects are expected from the first experiment. Serially balanced trials were developed originally for experiments where subjects could receive quite long sequences of treatments; more recently they have been used in agricultural field trials on fungicides where interference is expected between neighbouring plots.

We call the effect of a treatment on neighbouring units its residual or interference effect. If interference effects act in all directions equally they are called non-directional, if they act in one direction directional and in two directions unequally bi-directional. We restrict attention to one-dimensional designs i.e. those in which the interference effects act in one dimension only. In repeated measurement designs the dimension is time; in field experiments it is commonly determined by plot shape or direction of the prevailing wind etc.

Terms which are appropriate when the dimension is time tend to be inappropriate when the dimension is space. We try to be fairly consistent in our use of the following terms. A trial consists of several subjects each of whom receives a sequence of treatments. Each treatment in the sequence is applied for a given length of time called a period. The basic unit of measurement corresponds to a cell in the periods × subject table. Subjects are considered to be independent but may have a (block) structure superimposed on them. Subjects tested at the same time (or arranged in the same

direction) are called parallel.

We consider three types of design:

1) Change-over designs

2) Superimposed designs

3) Serially balanced designs.

The categories are neither exhaustive nor mutually exclusive. Let p denote the number of periods and t the number of treatments then typically the categories have the following properties:

Change-over designs

1) $p \leq t$

2) directional

3) subject structure: 'subject' or ' block/subject'

4) all cells of interest

Superimposed designs

1) $p = 2, 3$

2) directional

3) many subjects with simple block structure

4) only one period of interest.

Serially-balanced designs

1) $p = O(t^2)$ or $O(t^3)$

2) directional or non-directional

3) few subjects with no structure

4) all cells of interest.

Clearly occasions arise for designs in none of these categories but complete discussion is outside the range of this thesis.

Designs of this type do not satisfy standard randomization theory e.g. period 2 always follows period 1. In consequence no particular error model has been adopted as the overriding standard.

We consider two models

1) the errors in each cell are independently distributed with mean 0 and variance σ^2 and

2) (for serially balanced designs only) the errors x_i form a stationary Markov process $x_i = \rho x_{i-1} + \varepsilon_i$ where ε_i are independent identically distributed with mean zero and variance σ^2 .

We do not consider models with interactions between direct and residual effects.

6.3 Review of the literature

A useful bibliography of repeated measurements designs is given by Hedayat and Asfarinejad (1975).

Change-over designs

Much of the work on change-over designs has been based on the method adopted by Cochran et al (1941) in which constants are fitted, by the usual least squares procedure for direct and residual effects after removing differences between subjects and periods. The observation y_{ij} of the jth subject in the ith period is represented by

$$y_{ij} = m + p_i + s_j + t_{[i]} + r_{[i-1]} + e_{ij} \qquad 1)$$

where $[i]$ denotes the treatment applied in period i . The e_{ij} are independent with mean zero and variance σ^2 . This model is used extensively by other workers notably Williams (1949) and Patterson and Lucas (1962). Hedayat and Asfarinejad (1978) prove that some of the designs presented by the above authors are optimal under model 1) but this optimality is implicitly recognised in the earlier papers.

Patterson (1951) considers modifications to model 1) when there is a strong trend across periods. Finney (1956) considers several alternative error models and shows that a suitable model involving correlation and additive residual effects results in much the same analysis as one with a simple autoregressive scheme. Berenblut (e.g. 1970) in a series of papers considers change-over designs with special properties as does Patterson (1973).

The designs presented by Patterson and Lucas (1962) have the subjects arranged in complete blocks, balanced or partially balanced incomplete blocks; the number of periods, p is less than or equal to the number of treatments t. Davis and Hall (1969) present cyclic change-over designs for $p \leq t$ and subjects arranged in complete blocks. These designs are described in greater detail in §6.4.2.

Superimposed designs

Model 1) is suitable with minor modifications for use when a new experiment is superimposed on an existing one and the first set of treatments are assumed to have an additive residual effect. There need be no restriction that the two sets of treatments are the same. The construction and analysis of change-over and superimposed designs are therefore closely related. Pearce and co-workers (Hoblyn et al (1954), Freeman (1958)) have studied superimposed designs from the point of view of three non-interacting classifications. They categorise designs by the notation X:YZ where X denotes the relationship between the first two classifications and YZ the relationships between the third and the first two. Depending on which of the three classifications are regarded as block factors apparently different designs may be constructed. In particular if

any two categories are orthogonal the design may be derived from a row and column design. Some of these designs can be generated simply and there is further discussion in §6.5.2. Preece (1966, 1971) and Hedayat et al (1972) extend the above work to cover two non interacting treatments in Youden squares. Apart from a few special cases these designs do not appear suitable for automatic generation.

Serially balanced designs

Williams (1952) suggests using these designs when the errors in a one dimensional design are serially correlated in the form of a stationary first or second-order Markov process. He assumes there are no residual effects and presents designs in which each treatment is balanced for first neighbours when left and right neighbours are not distinguished; the treatments are arranged in complete blocks so that each treatment is evenly spread throughout the trial (but there is no adjustment for blocks). Finney and Outhwaite (1956) apply serially balanced designs to bioassay. Their approach differs from that of Williams in that they assume a directional residual effect, and a different error model; they also adjust for blocks. Sampford (1957) extends their work.

No serious attempt to enumerate serially balanced designs appears to have been made apart from that undertaken by Dyke and Shelley (1976) who use a computer to generate designs with four treatments balanced for pairs of immediate neighbours. They generate over 1500 distinct standard sequences but later (private communication) show there are 1566 in all. (In Appendix A6.6 we present 87 transformation sets each containing 18 standard sequences.)

Although randomization of serially balanced sequences is somewhat similar to that of Latin squares no corresponding randomization theory has been developed. (Under model 1) the designs are

not unbiased because choice of treatments in neighbouring plots from different blocks is not independent.) Recently other forms of analysis have been investigated. Jenkyn et al (1979) analyse data from fungicide trials using a four-term Fourier series as a covariate. Atkinson (1969) shows that the Papadakis analysis (Bartlett (1938)) is in practice very similar to that of Williams (1952).

Serially balanced designs are closely related to complete Latin squares and other two-dimensional designs which are discussed by Dénes & Keedwell (1974) and Freeman (1979) among others. These designs are of interest here only in that some of them can be generated by minor modifications to the procedures for serial balanced designs. The paper by Bartlett (1978) and the subsequent discussion highlight their advantages and disadvantages.

6.4 Change-over designs

6.4.1 Analysis

The construction and analysis of change-over designs is described in great detail by Patterson and Lucas (1962). For completeness we give an outline of the analysis of change-over designs when the structure of the subjects is ignored. The notation used is that of Davis and Hall (1969). There are n subjects, p periods and t treatments all replicated b times in each period. The model 1) in §6.3 can be re-expressed in matrix terms

$$\underline{y} = \mu E_{np} + D\underline{\delta} + R\underline{\rho} + U\underline{\nu} + P\underline{\pi} + \underline{\varepsilon} \qquad 1)$$

where E_{np} is a vector of length np with all elements unity; μ, $\underline{\delta}$, $\underline{\rho}$, $\underline{\nu}$ and $\underline{\pi}$ are the mean and vectors for direct, residual, subject and period effects respectively.

It is assumed that

$$\Sigma\delta_i = \Sigma\rho_i = \Sigma\nu_i = \Sigma\pi_i = 0 \ . \qquad 2)$$

The information matrix has the form

$$\begin{bmatrix} np & bpE^T_t & b(p-1)E^T_t & pE^T_n & nE^T_p \\ bpE_t & bpI_t & L & M & bJ_{t,p} \\ b(p-1)E_t & L^T & b(p-1)I_t & N & b\tilde{J}_{t,p} \\ pE_n & M^T & N^T & pI_n & J_{n,p} \\ nE_p & bJ^T_{t,p} & b\tilde{J}^T_{t,p} & J^T_{n,p} & n\,I_p \end{bmatrix} \qquad 3)$$

where $\tilde{J}_{t,p} = [0, J_{t,p-1}]$ and L,M,N are incidence matrices of direct and residual effects, direct and subject effects, residual and subject effects respectively. Under constraints 2) μ is estimated by the grand mean; the effects of periods and subjects are easily removed leaving reduced normal equations

$$C\begin{pmatrix}\hat{\delta}\\ \hat{\rho}\end{pmatrix} = \begin{bmatrix} \theta & \phi \\ \phi^T & \psi \end{bmatrix}\begin{bmatrix}\hat{\delta}\\ \hat{\rho}\end{bmatrix} = \begin{bmatrix} Q \\ S \end{bmatrix} \qquad 4)$$

where

$$\theta = bpI_t - p^{-1}NN^T \ , \quad \phi = L - p^{-1}NM^T \ , \quad \psi = b(p-1)I_t - p^{-1}MM^T \qquad 5)$$

$$Q = (D^T - p^{-1}NU^T)\underline{y} \ , \quad S = [R^T - p^{-1}MU^T - t^{-1}\{b^{-1}R^TPP^T - (1-p^{-1})J_{t,np}\}]\underline{y}$$

By and large the method of inversion of C is of minor concern; for none of the designs presented by Hall and Williams (1969) and only three of those presented by Patterson and Lucas (1962) does this matrix exceed 40×40. (As all three are also partially balanced special methods could be employed.) Where problems might occur two approaches can be made.

1) Use of designs presented by Patterson and Lucas results in incidence matrix L and concurrence matrices NN^T ,

NM^T and MM^T having a simple structure giving straightforward inversion.

2) Use of designs presented by Hall and Williams results in matrix C being of block circulant type. The inversion procedure given in Appendix A8.5 can be used.

For any suitable design C can be inverted by partitioning in the obvious way. For the extra-period change-over designs of Patterson and Lucas (1962) $\phi = 0$ and the inversion problem is trivial.

6.4.2 Construction

Designs presented by Patterson and Lucas (1962) and Davis and Hall (1969) provide a good standard collection and other designs are rarely required. In this section we outline how the designs of these authors may be generated within a general purpose design programme.

Cyclic Change-over designs (Davis and Hall, 1969)

These designs may be treated as special cases of cyclic designs (§2.4.1); as is pointed out in Chapter 2, cyclic generators although intended for a nested block structure can be used for other block structures. Suitable initial blocks for $6 \leq t \leq 20$, $3 \leq p \leq 5$ are given by Davis and Hall (1969).

General change-over designs (Patterson and Lucas, 1962)

Patterson and Lucas (1962) present 160 change-over designs. The construction procedures used can be reduced to combinations of a few basic operations.

One-stage procedures

1) Cyclic constructions (with variations) (31)

2) Ad-hoc construction (5)

3) Addition of extra period (65)

4) Deletion of period (25)

5) Compound of change-over and incomplete block design. (34)

A summary is given in Appendix A6.1.

Most designs in this set are small and all could be held in a small direct-access file. Possibly useful compromise procedures are i) store all designs apart from those obtained by adding or deleting periods and build these operations into the programs or ii) store generators for those with simple cyclic constructions and store complete designs otherwise.

6.5 Superimposed designs

6.5.1 Analysis

The model usually assumed for a superimposed design with p rows, n columns and t treatments in each period replicated b times in each row is

$$\underline{y} = \mu E_{np} + D\underline{\delta} + R\underline{\rho} + U\underline{\nu} + P\underline{\pi} + \underline{\varepsilon} \qquad 1)$$

where E_{np} is a vector of length np with all elements unity; μ, δ, ρ, ν and π are the mean and vectors for current (direct), residual, column and row effects respectively. This model corresponds to model 1) of §6.4.1 with rows substituted for periods, columns substituted for subjects but there are no residual effects in the first 'row' of the change-over design. There can be various modifications e.g. differing numbers of treatments in the two periods but the analysis of a superimposed design is very similar to that of a change-over design. By and large, the remarks in §6.4.1

on matrix inversion apply here also.

When both sets of treatments are equally replicated in all rows then the rows category is orthogonal to all other categories and can be eliminated. When the numbers of treatment in the two periods are not the same the analysis increases in complexity, only because the matrix C may have a more complicated form. However, in few designs does the dimension exceed 50 and there is little need to consider methods other than direct matrix inversion (after partitioning if necessary).

Efficiencies for contrasts of direct and residual effects may be determined in the usual way.

6.5.2 Construction

Perhaps the most useful collection of superimposed designs is presented by Hall and Williams (1973); some other constructions are also noted.

Cyclic superimposed designs (Hall and Williams, 1973)

These designs can be constructed using cyclic generators, (cf. the generation of cyclic change-over designs in §6.4.2). Suitable initial blocks for $t = 5$, $k = 4$ and $6 \leq t \leq 15$, $k = 4, 5, 6$ are to be found in Hall and Williams (1973).

Other designs

The model 1) has four additive factors; designs presented in the literature usually have three or four factors. Three factor designs can be constructed by suppressing one factor in four factor designs. Also when one or more pairs of factors are orthogonal the designs can be constructed as row and column designs. Various designs for three or four classifications have been published (Freeman (1958), Pothoff (1963), Causey (1968)) but do not, by and

large, land themselves to construction by simple generators. Obvious exceptions are Graeco-Latin squares; some closely related constructions are noted in Appendix A6.2.

As with change-over designs, superimposed designs tend to be small and apart from those which have straight-forward constructions most others could be easily stored complete in a direct-access file.

6.6 Serially balanced designs

6.6.1 Definitions and analysis

A variety of models have been used in the analysis of serially balanced designs. These models vary in two main respects:

i) the interference effects

ii) the error model.

In this section we restrict attention to two basic models, see analyses A and B below. In both cases the design is one-dimensional and each treatment has a direct effect on a cell to which it is applied and an additive interference effect on the one following (e.g. its right neighbour).

We consider only designs based on a single sequence and because of the one-sided interference effect assumed we do not consider non-directional designs for n replicates of $2n$ or $2n+1$ treatments such as those presented by Williams (1952). We define three types of sequence each comprising a set of complete blocks (the basic sequence) but with extra treatments added as necessary at either end to remove end effects. The three types are:

Type I

The basic sequence has length kt^2 with t symbols arranged in kt complete blocks. All t^2 ordered pairs of successive

symbols occur k times. (k is called the index.)

Type II

The basic sequence has length $kt(t-1)$ with t symbols arranged in $k(t-1)$ complete blocks. All $t(t-1)$ ordered pairs of successive different symbols occur k times.

Type III

These designs are balanced for triplets of treatment symbols and three sub-types are considered .

Type IIIa the sequence is of type I with $k = c(t-1)$. All $t^2(t-1)$ possible ordered triplets of successive symbols occur c times. (Triplets xxx and yxy cannot occur.)

Type IIIb the sequence is of type II with $k = c(t-1)$. All $t(t-1)^2$ ordered triplets of successive symbols xyz with $x, z \neq y$ occur c times.

Type IIIc the sequence is of type II with $k = c(t-2)$. All $t(t-1)(t-2)$ possible ordered triplets of successive different symbols occur c times.

Example 6.1

i) Type I $t = 3, k = 2$; properties i) and ii) (see §6.6.2)
123 312 231 132 213 321 .

ii) Type II $t = 5, k = 1$; properties ii), iii) and iv)
12345 24135 31425 43215 .

iii) Type IIIa example i) is also a type IIIa sequence.

iv) Type IIIb $t = 4, c = 1$; (also type II, $k = 3$)
1234 3421 3124 2431 3412 1423 2413 2314 1432.

v) Type IIIc $t = 4, c = 1$; (also type II, $k = 2$); properties ii) and iii)
1234 2314 2134 1324 3214 3124 .

Type I and type II designs are special cases of a more general class in which each pair of ordered symbols xy occurs k_1 times and each pair of like symbols xx occurs k_2 times. The number of blocks for these designs is $k_2 + (t-1)k_1$ or $k_2 t + (t-1)(k_1 - k_2)$. (For designs arranged in complete blocks k_1 is never less than k_2.) For type I designs $k_1 = k_2 = k$ and for type II designs $k_1 = k$, $k_2 = 0$.

In analysis B presented below and appendix A6.3 we show that designs which have properties of balance for next-but-one neighbours are especially useful. Type IIIa and Type IIIc designs have the Type II property with respect to next-but-one neighbours while Type IIIb designs have the more general property - ordered pairs xy occur $c(t-2)$ times and like pairs xx occur $c(t-1)$ times. These designs are also useful in the absence of interference effects when the errors follow a second-order autoregressive process (Williams (1952)).

Analysis A

We assume the model

$$y_{ij} = m + b_j + a_{[i]} + s_{[i-1]} + \varepsilon_{ij} \qquad 1)$$

where y_{ij} is the yield of the ith cell of the jth block, b_j is the effect of the jth block, $a_{[i]}$ is the direct effect of the treatment applied to that cell and $s_{[i-1]}$ is the interference effect from the previous cell. The direct treatment and block effects are orthogonal and hence the design may be analysed as a row and column design with direct effects and blocks corresponding to the rows and columns and interference effects corresponding to treatments. In all the designs considered here the total number of parameters is small and there is no need to consider any special

matrix inversion techniques. We observe one special case in which the same symbol occupies the last cell in each block (called a totally reversible sequence by Finney and Outhwaite (1956)). These sequences, which must be of type II correspond to O : OT designs in the notation of Pearce (1963).

Analysis B

Many of the serially balanced designs considered here are sufficiently long that they may be analysed with the help of the important methods of time series analysis, e.g. spectral analysis or moving average/autoregressive models. We restrict attention to autoregressive models with particular attention to the first order (Markov process) model. Consider a design with $n = m + p$ cells with the errors obeying a pth order autoregressive model

$$z_i + \Sigma d_s z_{i-s} = \varepsilon_i \qquad s = 1 \ldots p$$

where d_s satisfy appropriate conditions for stationarity, and ε_i are independent identically distributed with mean zero and variance σ^2. The joint distribution of the $m + p$ observations can be expressed as the joint distribution of the first p observations and the ε_i, (i.e. ignoring mean and treatment effects)

$$p(\underline{z}|\underline{d}, \sigma^2) = p(z_{p+1},\ldots,z_n|\underline{z}_p, \underline{d}, \sigma^2). \quad p(\underline{z}_p|\underline{d}, \sigma^2)$$

$$= p(\varepsilon_{p+1},\ldots,\varepsilon_n, \sigma^2). \quad p(\underline{z}_p|\underline{d}, \sigma^2) . \qquad 1)$$

For small values of p (relative to n) a simple and relatively efficient procedure for estimating the model parameters is to ignore the contribution of the first p cells except in so far as they help to give estimates for $\varepsilon_{p+1},\ldots,\varepsilon_n$, i.e. to maximise the likelihood based on the first term of expression 1). For a serially balanced design with (left) interference effects but no block effects and a first-order autoregressive model, the logarithm of the

likelihood based on the conditional distribution is

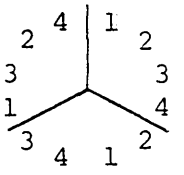
$$-\frac{1}{2\sigma^2}\Sigma((y_i-m-a_{[i]}-s_{[i-1]}) - d_1(y_{i-1}-m-a_{[i-1]}-s_{[i-2]}))^2$$

where $a_{[i]}$, $s_{[i]}$ are direct and interference effects of the treatment applied to the ith cell respectively. In general, solution of the maximum likelihood equations involves the inversion of a $2(t-1) \times 2(t-1)$ matrix to derive estimates of the a_q and s_q ; $(q = 1,2,\ldots,t)$ estimates of d_1 are derived from the correlations of residuals on neighbouring cells. More details are given in Appendix A6.3.

6.6.2 Construction

We consider only designs in which treatments are arranged in complete blocks. These designs guard against large trends but also, by being essentially robust, allow for the use of different models in the analysis of data coming from the associated trials.

Serially balanced designs are closely related to Latin squares and it is therefore desirable that facilities for selecting designs at random should be available. All the sequences considered in this chapter are essentially circular and consist of b complete blocks of t treatments, which may be written round a circle e.g.



The circle can be broken between any pair of blocks and the sequence read in either direction; end cells are added as necessary to provide the appropriate residual effects. We define a standard sequence as one in which treatments of the first block are ordered 1, 2 ... t . Each circle provides at most 2b standard sequences

each of which provide t! sequences through permutation of the symbols. Finney and Outhwaite (1956) define all sequences derivable from a given circle as belonging to the same transformation set.

In Appendices A6.4 - A6.6 standard sequences for Type I, II, IIIa, IIIb, IIIc designs are given. Only one member of any transformation set is presented together with the number of distinct standard sequences in the transformation set. To save space, attention has been restricted to sequences with no more than 60 cells or no more than 12 replications. The sequences presented may be summarised by the following table:

		number of treatments					
	blocks	t=3	4	5	6	7	8
Type I	t	*	*	*	a	s	-
II	t-1	a	a	s	s	s	s
IIIa	t(t-1)	a	a	-	-	-	-
IIIb	$(t-1)^2$	a	a	-	-	-	-
IIIc	(t-1)(t-2)	a	a	s	-	-	-

where a and s indicate all and some of the standard sequences are presented; * indicates that no sequence exists and the sequences presented are for index k = 2, or 3. Where a selection of sequences are presented they are usually chosen on the grounds of some useful property. The properties looked for were:

i) sequence balanced for next-but-one neighbours

ii) sequence nearly balanced for block and residual effects

iii) sequence totally reversible

iv) sequence based on Latin square with an extra column containing one treatment only. (The sequence is read by rows.)

Sequences with property iv) normally also have property ii) or iii). Sequences with property i) are useful with the autoregressive model, those with property ii) for block models. Those with property iii) are also useful for block models and for constructing other sequences by suitable block permutation and cycling of the symbols within certain blocks. Sequences formed from reading clockwise can be very different with respect to property ii) from those in the same transformation set read anticlockwise.

Example 6.2

The totally reversible type II sequence

12345 24135 31425 43215

satisfies properties ii), iii) and iv). Permutation of blocks yields another totally reversible sequence, Cycling, say, block 2 to give block 35241 yields another type II sequence.

The sequences presented in Appendices A6.4 - A6.6 were obtained in the main through efficient computer search routines. The routines which employ back-tracking techniques, if necessary, were designed to be easily modified to suit different requirements. The search techniques normally yielded all standard sequences. Where a large number of sequences were generated e.g. the type IIIb designs for $t = 4$ the following simple technique proved useful in both determining which sequences come from the same transformation set and ordering the transformation sets:

> Determine the mean position m_i of treatment i (counting from the left). Derive $S_2 = \Sigma(m_i - \bar{m})^2$, $S_3 = \Sigma(m_i - \bar{m})^3$. Sequences in the same transformation set have equal value of S_2 and the same absolute value of S_3. (If $S_3 \neq 0$ the sign indicates the direction of reading.)

Transformation sets with small values of S_2 and S_3 are often desirable while those based on totally reversible designs have high values. Where appropriate, the transformation sets given in Appendices A6.4 to A6.6 have been ranked in order of S_2 and S_3 and the values printed alongside.

Appendix A6.1

Constructions for the change-over designs of Patterson & Lucas (1962)

Methods for constructing 160 change-over designs given by Patterson and Lucas are summarised. They fall into four categories:

i) producing a basic design (method C)

ii) amending a design by the addition of an extra period or deletion of one or more periods (methods E, D)

iii) compounding a change-over and a block design (method P)

iv) ad-hoc methods.

Methods

Method C This method covers two types of cyclic generation.

C1. The subjects are grouped into b blocks each generated cyclically from its own initial block.

C2. There are $b = kc$ blocks, k blocks are specified and c blocks are generated cyclically from each of these.

Method E An extra period is added by repeating the pth period.

Method D One or more periods are deleted from a design arbitrarily, D1, or selectively, D2.

Method P Construct two designs:

A - a BIB or PBIB design for t treatments in blocks of k subjects;

B - change-over design for k treatments in blocks of k_2 subjects. Construct B for each block of A; the resulting design has t treatments and $b = b_1 b_2$ blocks of k_2 subjects (see §7.2.4).

Method		Designs
C	C1	2, 5, 10, 11, 14, 15, 19, 21, 23, 25, 26, 27, 28
	C2	59, 63, 65, 68, 71, 72, 73, 94...98, 126, 153, 154, 156, 159, 160
E		30, 31...54, 76, 77...93, 131, 132...152
D	D1	1, 3, 4, 7, 8, 9, 16, 18, 20, 22, 55, 57, 58, 61, 62, 64, 66, 67, 125, 127
	D2	12, 13, 17, 24, 70
P		29†, 69, 74, 75, 99, 100...124, 128, 129, 130, 157
Ad-hoc		6, 56, 60, 155, 158

† design 74 with one block stratum ignored.

Note i) Many designs labelled D and E can be generated cyclically.

ii) Some designs labelled ad-hoc have generalized cyclic generators e.g. design 158 for t = 10, p = 3, k = 2, b = 15 can be generated with increment 2 from initial blocks:

0 4 0 5 1 3

4 0 5 0 3 1

4 0 5 0 3 1

Appendix A6.2

Simple constructions for superimposed designs with four factors

We present as a theorem some useful general methods for constructing designs with four factors when the number of levels differ by no more than 1. (Most results are not original.) The notation used is that of Pearce (1963).

Theorem Let A and B denote Latin squares of order t, A' and B' denote designs derived from them and (t_1,t_2,t_3,t_4) denote the number of levels of the four factors in the derived designs, then

i) If A and B are orthogonal with a common transversal (e.g. they form a pair of COLS (§5.2)) then there exists an O:OO:SSS design for levels $(t,t,t,t+1)$.

ii) If A and B satisfy the same conditions as in i) then there exists an O:TO:OTO design for levels $(t,t+1,t,t+1)$.

iii) The existence of p MOLS (§5.2) of order t implies the existence of p mutually orthogonal Youden squares of order $(t-1)\times t$ i.e. each pair forms an O:OT:OTT design for levels $(t-1,t,t,t)$.

iv) For all t a pair of orthogonal $(t-1)\times t$ Youden squares can be constructed cyclically.

v) For all t there exists an O:OO:SSS design for levels $(t,t,t,t-1)$.

Proof The proofs are given by construction.

i) Form B' by replacing the common transversal of B by added treatment x. Superimpose B' on A.

ii) Form A' and B' as copies of A and B but with an extra column containing a copy of the common transversal. Replace the common transversal in B' by the extra treatment

x. Superimpose B' on A' .

iii) Write all the Latin squares with a standard first row then delete the first row. In particular, orthogonal Latin squares constructed from design key matrices of the form $[C_i, I]^T$ can be used.

iv) Note that in iii) orthogonal Latin squares can be constructed by cyclic addition of 1 from the first columns

A $(0,1,t-1,2,t-2,\ldots,r,r+1,t-r,r+2,\ldots,m+1)^T$

B $(0,t-1,2,t-2,\ldots,r,r+1,t-r,r+2,\ldots,m+1,1)^T$

where $t = 2m+1 = 4r\pm1$. For even values of t , near orthogonal Latin squares can be constructed from the first columns

A $(0,1,t-1,2,t-2,\ldots,t/2)^T$

B $(t/2,0,1,t-1,2,\ldots)^T$.

In both cases deletion of the first rows provides an orthogonal pair of Youden squares A' and B' or order $(t-1)\times t$.

v) Construct Latin squares A and B as for iv). Note that for t odd, treatment $t-1$ in B occurs with each treatment in A ; for t even treatment $t-1$ in B occurs with treatment $t/2-1$ in A twice, $t-1$ not at all and all other treatments once. Form B' by replacing treatment $t-1$ in B by the treatment in the equivalent cell of A ; if this cell contains $t-1$ then substitute treatment zero. Superimpose B' on A .

The above constructions can be derived readily form others found useful elsewhere in this thesis. Thus, for example, construction i) requires substituting an invariant treatment in the leading diagonal of a twin (§5.2).

Appendix A6.3

Approximate Maximum Likelihood estimates for serially balanced designs

We assume that the errors follow a linear autoregressive process, in particular, the first-order Markov process and provide an approximation to the maximum likelihood solution where each treatment has a direct effect on the plot or cell to which it is applied and an additive interference effect on the following cell (its right neighbour). Other approximations to autoregressive processes are given by Box and Jenkins (1976, Appendix A7.5).

We assume that the sequence consists of $n+2$ cells numbered $-1,0,1,\ldots,n$ where cells -1 and 0 have the same treatments as plots $n-1$ and n respectively. Cell -1 is not measured and each treatment is applied to r of the cells $1,2,\ldots,n$.

Let the errors for cells $0,1,\ldots,n+p-1$ follow a pth order autoregressive process

$$z_i + \Sigma d_s z_{i-s} = \varepsilon_i \qquad (s = 1 \ldots p)$$

where the d_s satisfy the appropriate conditions for stationarity and the ε_i are independent identically distributed with mean zero and variance σ^2. The distribution of the z_i can be expressed in terms of the joint distribution of the first p values $z_0,\ldots,z_{p-1}$ and the next n ε values.

$$p(\underline{z}|\underline{d},\sigma^2) = p(z_p,\ldots,z_{n+p-1}|\underline{z}_p,\underline{d},\sigma^2).p(\underline{z}_p|\underline{d},\sigma^2)$$

$$= p(\varepsilon_p,\ldots,\varepsilon_{n+p-1},\sigma^2).p(\underline{z}_p|\underline{d},\sigma^2) \qquad 1)$$

If information from the first p cells is ignored, other than for determining the $\varepsilon_i (i \geq p)$, then an approximate likelihood function can be based on the distribution of the ε_i in expression 1). This approximation is reasonable for p small relative to n.

For the first order model the log likelihood function is

$$L_1 = K - n \log \sigma - \frac{1}{2\sigma^2} \sum_1^n [(y_i - m - a_{[i]} - s_{[i-1]}) - \rho(y_{i-1} - m - a_{[i-1]} - s_{[i-2]})]^2 \qquad 2)$$

where y_i is the yield of the ith cell, m is the overall mean, $a_{[i]}$ is the direct effect of the treatment applied to the ith cell and $s_{[i]}$ is the interference effect of the treatment applied to the ith cell. A second approximate likelihood function can be obtained by reversing the autoregressive process and using the distribution conditional on the last cell.

$$L_2 = K - n \log \sigma - \frac{1}{2\sigma^2} \sum_1^n [(y_{i-1} - m - a_{[i-1]} - s_{[i-2]} - \rho(y_i - m - a_{[i]} - s_{[i-1]})]^2 \qquad 3)$$

Estimates derived from L_1 and L_2 are highly correlated but average estimates take a simple form.

Let suffices O, 1 denote summation between O and n-1 or between 1 and n respectively. Define T_O, T_1 to be totals of cells O ... n-1 and 1 ... n respectively; T_{Oq}, T_{1q} to be totals over cells containing treatment q, and L'_{Oq}, R'_{1q}, R'_{Oq}, R''_{1q} to be totals of cells one to left, one to right and two to right of cells containing treatment q. The maximum likelihood equations for L_1 are

$$\sigma^2 \frac{\partial L_1}{\partial m} = T_1 - \hat{\rho} T_O - (1-\hat{\rho})\, n\hat{m} - r(1-\hat{\rho}) \sum \hat{a}_q - r(1-\hat{\rho}) \sum \hat{s}_q \qquad 4)$$

$$O = \sigma^2 \frac{\partial L_1}{\partial a_q} = (1+\hat{\rho}^2)\{\bar{T}_q - r\hat{m} - r\hat{a}_q - L'_q(\hat{s})\} - \hat{\rho}\{L'_{Oq} - r\hat{m} - L'_q(\hat{a}) - L''_q(\hat{s})\}$$
$$- \hat{\rho}\{R'_{1q} - r\hat{m} - R'_q(\hat{a}) - r\hat{s}_q\} \qquad 5)$$

$$O = \sigma^2 \frac{\partial L_1}{\partial s_q} = (1+\hat{\rho}^2)\{\bar{R}'_q - r\hat{m} - R'_q(\hat{a}) - r\hat{s}_q\} - \hat{\rho}\{T_{Oq} - r\hat{m} - r\hat{a}_q - L'_q(\hat{s})\}$$
$$- \hat{\rho}\{R''_{1q} - r\hat{m} - R''_q(\hat{a}) - R'_q(\hat{s})\} \qquad 6)$$

where $(1+\hat{\rho}^2)\,\bar{T}_q = \{T_{1q} + \hat{\rho}^2 T_{Oq}\}$ and $(1+\hat{\rho}^2)\,\bar{R}' = \{R'_{Oq} + \hat{\rho}^2 R'_{1q}\}$; $L'_q(\hat{s})$ is the total of s effects from plots one to left of plots containing treatment q ; $L''_q(\hat{s})$, $R'_q(\hat{s})$, $L'_q(\hat{a})$, $R'_q(\hat{a})$, $R''_q(\hat{a})$, $L''_q(\hat{s})$ are defined similarly. The equation derived from L_2 differs only in that T_{Oq} , T_{1q} are interchanged in the expressions for $\bar{T}_1$ and similarly for $\bar{R}'$. Averaging over the L_1 , L_2 equations gives

$$\hat{m} = \frac{T_O + T_1}{2n}$$

and $$\bar{T}_q = \frac{T_{Oq} + T_{1q}}{2} , \qquad \bar{R}'_q = \frac{R'_{Oq} + R'_{1q}}{2}$$

If the design is balanced for neighbours then expressions 5) and 6) simplify. They simplify further if the design is also balanced for next-but-one neighbours. Let each treatment occur as left neighbour to itself g_1 times and to each other treatment g_2 times; also let each treatment occur as left-but-one neighbour to itself h_1 times and to each other treatment h_2 times. Write $g = g_2 - g_1$, $h = h_2 - h_1$ and set $\Sigma a_q = 0$, $\Sigma s_q = 0$ then equations 5) and 6) become

$$C_1\hat{a}_q + C_2\hat{s}_q = (1+\hat{\rho}^2)(\bar{T}_q - r\hat{m}) - \hat{\rho}((L'_{Oq} - r\hat{m}) + (R'_{1q} - r\hat{m})) \qquad 7)$$

$$C_2\hat{a}_q + C_1\hat{s}_q = (1+\hat{\rho}^2)(\bar{R}'_q - r\hat{m}) - \hat{\rho}((T_{Oq} - r\hat{m}) + (R''_{1q} - r\hat{m})) \qquad 8)$$

where $C_1 = (1+\hat{\rho}^2)r + 2g\hat{\rho}$

$C_2 = \hat{\rho}(h-r) - g(1+\hat{\rho}^2)$

Equations 7) and 8) are effectively the least squares equations and estimates and their variances can be obtained in the usual way tbrough inversion of matrix $X'X = \begin{bmatrix} C_1 & C_2 \\ C_2 & C_1 \end{bmatrix}$. Estimates for $\hat{\rho}$ are derivable from both likelihood expressions L_1 and L_2 :

$$\hat{\rho} = \frac{\sum_{i=0} z_i z_{i+1}}{\sum_{i=0} z_i^2} \quad \text{or} \quad \hat{\rho} = \frac{\sum_{i=0} z_i z_{i+1}}{\sum_{i=1} z_i^2} \; .$$

A simple average is derivable by replacing z_0^2 or z_n^2 in the denominator by $\left[\frac{z_0 + z_n}{2}\right]^2$ or $\frac{z_0^2 + z_n^2}{2}$.

Appendix A6.4

Representatives of type I designs

Representative standard sequences for type I designs are presented together with the number of standard sequences which they produce. For $t \leq 5$ no type I design of index 1 exists and representative designs of index 2 are given. For $t = 6$ representatives are given for all transformation sets together with the values of S_2 and S_3 (§6.6.2). Sampford (1957) presents sequences for $t > 7$. (* Sequence balanced for next-but-one neighbours.)

t = 3 Index 2

123 312 231 132 213 321 (sequences = 3) *

t = 4 Index 2

1234 4132 2413 3421 1243 3142 2314 4321 (4)

t = 5 Index 2

12435 51324 45213 34152 23541 -
-13452 24513 35124 41235 52341 (20)

t = 6 Index 1 (12 sequences unless shown).

						S_2	S_3
123456	615432	253164	462135	514263	365241	56	24
123456	615243	351642	265314	413625	546321	56	24
123456	652143	315462	264135	516324	425361	72	0
123456	641325	514362	263154	421653	352461	88	72
123456	641325	531462	216543	361524	426351	112	144
123456	614352	246315	513264	416253	365421	112	144
123456	654132	216435	514263	361524	462531	120	120
123456	642153	316524	461325	514362	263541	120	120
123456	615432	241365	514263	352164	462531	128	72
123456	613524	431625	514632	264153	365421	128	72
123456	621543	361425	513264	416352	246531	144	0
123456	613254	415263	351462	243165	536421	144	48
123456	652143	316425	513624	461532	263541	144	48
123456	624153	316425	514632	213654	435261	152	24
123456	641532	213654	431625	514263	352461	152	24
123456	631524	426135	516432	214653	362541	152	120
123456	613524	426315	516432	214653	362541	152	120
123456	654132	215364	431625	514263	352461	152	360
123456	624153	316425	514632	261354	436521	152	360
123456	621543	316425	532614	413652	246351	152	408

						s_2	s_3	
123456	614253	315264	462135	541632	243651	160	72	
123456	652143	315462	241635	513264	425361	184	288	
123456	613542	264153	314625	516324	436521	200	48	
123456	613524	432165	514263	315462	253641	216	168	
123456	613524	425163	315462	214365	532641	216	216	
123456	614253	315462	241635	521364	432651	232	144	
123456	624135	536142	216543	315264	463251	112	144	(6)
123456	614253	315264	432165	541362	246351	208	288	(6)

t = 7 Index 1

1234567 7132465 5142736 6354172 2615743 3752164 4762531 (14)

Appendix A6.5

Representatives of type II designs

Representative type II sequences with index 1 are presented. (One sequence of index 2 is presented for t=5, others for t = 3,4 are given in Appendix A6.6.) For t = 3,4,5 complete catalogues are given; for t = 6,7,8 representative sequences are given. Marked sequences have special properties:

* sequence is balanced for next-but-one neighbours;

† sequence is based on Latin square with extra column;

TR sequence is totally reversible.

t = 3

123 132 (sequences = 2) * † TR

t = 4

1234 2143 1324 (6)

1234 3214 2413 (6)

1234 3241 4213 (2)

t = 5 (8 sequences unless shown)

	S_2	S_3		
12345 14325 35421 52413	8	0	(4)	
12345 35421 52413 14325	8	0	(4)	
12345 25431 35142 41532	8	0		
12345 43521 42513 15324	16	0	(4)	
12345 41532 52431 35142	24	24	(4)	
12345 35421 43251 52413	48	96		
12345 35241 43251 54213	48	96		
12345 35241 54321 42513	48	96		
12345 42513 15324 35214	56	72		
12345 31524 25413 51432	80	0		
12345 42531 43215 13524	80	240	(4)	
12345 25143 53241 31542	80	240	(2)	†
12345 32514 35421 52413	88	72		
12345 32514 35241 54213	88	72		
12345 43251 35241 42153	88	72		
12345 35241 43215 42513	88	72		
12345 43215 35241 42513	88	72		
12345 25431 53241 35142	88	72		
12345 14352 42531 32154	88	216		
12345 43251 42153 13524	88	216		
12345 32415 13542 52143	96	24		
12345 32541 35142 15243	96	192	(4)	

				S_2	S_3			
12345	32154	14352	42513	104	144			
12345	43215	13524	25314	120	120			
12345	42153	13524	32514	128	72			
12345	32415	25143	13542	128	72			
12345	24153	51432	54213	128	456			
12345	42153	51432	52413	128	456			
12345	42513	52143	15324	136	144			
12345	35214	25431	51324	136	432			
12345	25314	35421	51324	136	432			
12345	32541	43152	42135	136	432			
12345	24315	35421	41325	136	432			
12345	35421	41325	24315	136	432			
12345	42513	21435	31524	144	48	(4)		
12345	14352	42153	25413	168	192			
12345	41532	42135	25143	168	192			
12345	41352	42153	25143	168	192			
12345	14325	42153	52413	168	192			
12345	14325	24153	54213	168	192			
12345	13254	35214	24153	176	144			
12345	35241	32514	21543	184	216			
12345	35241	32154	25143	184	216			
12345	31425	24135	15432	200	0			
12345	24135	31425	15432	200	0			
12345	15243	25314	13542	208	0			
12345	13542	15243	25314	208	0			
12345	35214	13254	24315	216	264	(4)		
12345	35214	24315	13254	216	264			
12345	32541	31524	21435	216	264			
12345	24153	25143	54213	248	360			
12345	25143	54213	24153	248	360			
12345	25143	24153	54213	248	360			
12345	24153	54213	25143	248	360			
12345	42153	52413	25143	248	360			
12345	42153	25143	52413	248	360			
12345	31524	21435	41325	256	360			
12345	41325	31524	21435	256	360			
12345	32514	21543	13524	264	528			
12345	32154	13524	25143	264	528			
12345	32154	25143	13524	264	528			
12345	32514	13524	21543	264	528			
12345	21543	24135	31425	280	360			
12345	31425	21543	24135	280	360			
12345	13524	21543	25314	280	360	(4)		
12345	24135	43251	42153	288	384			
12345	43251	42135	24153	288	384			
12345	24135	32154	31425	320	1080			
12345	32154	31425	24135	320	1080			
12345	24135	31425	43215	320	1920		TR	†
12345	24135	43215	31425	320	1920	(2)	TR	†
12345	31425	43215	24135	320	1920	(2)	TR	†
12345	32514	13524	31542	328	144			
12345	32514	31542	13524	328	144			
12345	31542	13524	32514	328	144			
12345	31524	32514	13542	328	144			
12345	32514	31524	13542	328	144			

				S_2	S_3	
12345	31524	32514	21354	368	1008	
12345	32514	31524	21354	368	1008	
12345	32514	21354	31524	368	1008	
12345	31524	21354	32514	368	1008	
12345	31524	13254	21435	376	72	(4)
12345	13254	21435	31524	376	72	
12345	21543	24135	14253	400	0	

Index 2

12345 14253 24135 31542 21345 32415 23514 31254 (8) *

t = 6

123456 326514 624135 425361 521643 (2) * †

123456 314652 436251 642153 261354 (2) †

123456 315462 536421 652413 261435 (2) †

123456 265413 631425 352461 516432 (2) †

123456 546213 635142 432615 253164 (10)

123456 243615 465213 531642 632514 (10)

t = 7

1234567 1642753 6135724 6251473 2637415 2176543 (4) *

1234567 1357246 1473625 1526374 1642753 1765432 (12) † TR

1234567 1352747 1426375 1573624 1647253 1765432 (6) † TR

t = 8

12345678 24631875 43862571 36584172 -
- 68153274 85216473 51428376 (2) †

12345678 71324658 51427368 63541728 -
- 26157438 37521648 47625318 (14) TR

Appendix A6.6

Representatives of type III sequences

Representatives of type IIIa, IIIb and IIIc sequences are given for t = 3 and 4 symbols. For t=5 one totally reversible sequence is given which can be used to construct others by permuting blocks and by some permutations of symbols within blocks.

Type IIIa

t = 3 (Type I, index 2)

123 312 213 321 132 231 (sequences = 3)

t = 4 (Type I, index 3)

1234 4132 2143 3124 4213 3241 1342 2314 4321 1424 3412 2431 (24)

1234 4132 2143 3124 4213 3241 1342 2431 1423 3412 2314 4321 (24)

1234 4132 2143 3124 4213 3241 1423 3412 2314 4321 1342 2431 (24)

1234 4132 2143 3124 4213 3241 1423 3412 2431 1342 2314 4321 (24)

1234 4132 2143 3124 4213 3412 2431 1342 2314 4321 1423 3241 (24)

1234 4132 2143 3124 4321 1342 2314 4213 3241 1423 3412 2431 (24)

1234 4132 2143 3241 1342 2431 1423 3412 2314 4213 3124 4321 (24)

1234 4132 2143 3241 1423 3124 4213 3412 2431 1342 2314 4321 (24)

1234 4132 2143 3241 1423 3412 2431 1342 2314 4213 3124 4321 (24)

1234 4132 2314 4213 3124 4321 1423 3241 1342 2143 3412 2431 (24)

1234 4132 2143 3124 4213 3412 2431 1423 3241 1342 2314 4321 (8)

1234 4132 2314 4321 1342 2143 3412 2431 1423 3124 4231 3241 (8)

Type IIIb

t = 3 (Type II, index 2)

123 132 321 213 (4)

123 132 321 312 (4)

t = 4 (Type II, index 3) (All yield 18 standard sequences)

									S_2	S_3
1234	3421	3124	2431	3412	1423	2413	2314	1432	12	12
1234	3421	3412	1423	2431	3124	2413	2314	1432	12	12
1234	3214	1243	4231	3142	4213	4132	3241	4312	20	0
1234	3241	4312	1243	4231	3142	4213	4132	3214	20	0
1234	2431	3142	3241	4132	3124	2143	4321	2134	44	36
1234	2431	4132	3241	4231	3124	2143	4321	3412	68	144
1234	2143	4321	3124	2431	4132	3241	4231	3412	68	144
1234	3241	4213	4231	3124	2431	4132	3214	3412	76	108
1234	2431	4132	3241	4321	4231	3124	2134	3412	76	108
1234	2431	4321	4132	3241	4231	3124	2134	3412	76	108
1234	3214	1243	4132	3142	4213	1342	3241	4312	84	96
1234	3241	4312	1243	4132	3142	4213	1342	3214	84	96
1234	3214	3412	4231	4132	3241	4213	1342	4312	108	60
1234	3241	4132	3214	3412	4231	4213	1342	4312	108	60
1234	3241	4231	4132	3214	3412	4213	1342	4312	108	60
1234	3241	4213	1342	4231	4132	3214	3412	4312	108	60
1234	3241	4213	1342	4312	4231	4132	3214	3412	108	60
1234	3412	4231	4132	3214	3241	4213	1342	4312	108	60
1234	3241	4132	3214	3412	4213	1423	1342	4312	108	324
1234	3241	4132	3214	3421	2431	3142	4231	2134	116	0
1234	3421	4231	3124	2431	4132	3241	4321	2134	116	288
1234	3421	4132	3241	4231	3124	2431	4321	2134	116	288
1234	3421	4132	3241	4321	2431	3142	4231	2134	116	288
1234	3142	4132	3214	1432	4213	4231	3124	3412	116	288
1234	3142	4213	4231	3124	3412	1432	3241	4132	116	288
1234	3241	4132	3214	3412	4213	4231	3142	4312	148	0
1234	2134	3412	4231	3142	4321	4132	3241	4312	148	0
1234	3412	4132	3214	1432	4213	4231	3142	4312	180	480
1234	3412	4213	4231	3142	4132	3214	1432	4312	180	480
1234	3214	3412	4213	1342	3241	4132	3142	4312	204	300
1234	3241	4132	3214	3412	4213	1342	3142	4312	204	300
1234	3241	4132	3142	4312	4213	1342	3214	3412	204	300
1234	3241	4132	3142	3214	3412	4213	1342	4312	204	300
1234	3241	4132	3142	4213	1342	3214	3412	4312	204	300
1234	3412	4213	1342	3214	3241	4132	3142	4312	204	300
1234	3142	4213	1342	3241	4321	4132	3124	3412	204	300
1234	3412	4213	4231	3142	4312	1432	3241	4132	212	288
1234	3421	4132	3241	4321	3124	2431	4231	3412	236	252
1234	3421	3124	2431	4321	4132	3241	4231	3412	236	252
1234	3421	3124	2431	4132	3241	4321	4231	3412	236	252
1234	3421	4231	3124	2431	4132	3241	4321	3412	236	252
1234	3421	4321	3124	2431	4132	3241	4231	3412	236	252
1234	3421	4231	4132	3241	4321	3124	2431	3412	236	252
1234	3421	4132	3241	4231	3124	2431	4321	3412	236	252
1234	3421	4132	3241	4231	4321	3124	2431	3412	236	252
1234	3241	4132	3214	3412	4231	3142	1342	4312	244	0
1234	3241	4132	3214	3412	1342	4312	4231	3142	244	0
1234	3241	4132	3214	3412	4312	1342	4231	3142	244	0
1234	3241	4132	3214	3412	4231	3142	4312	1342	244	0
1234	3412	4213	1342	3241	4132	3142	4321	4312	268	756
1234	3412	4213	1342	4321	4132	3142	3241	4312	268	756
1234	3412	4321	4132	3142	4213	1342	3241	4312	268	756

									S_2	S_3
1234	3412	4213	1342	3241	4321	4132	3142	4312	268	756
1234	3412	4213	1342	3142	4321	4132	3241	4312	268	756
1234	3421	4132	3241	4321	3142	4231	2431	3412	276	192
1234	3421	3412	4132	3214	1432	4231	3142	4312	276	672
1234	3421	3142	4132	3214	1432	4231	3412	4312	276	672
1234	3421	3142	4132	3214	1432	4312	4231	3412	276	672
1234	3421	3412	4231	3142	4132	3214	1432	4312	276	672
1234	3421	3142	4231	3412	4132	3214	1432	4312	276	672
1234	3421	3142	4312	4132	3214	1432	4231	3412	276	672
1234	3412	4231	3421	3142	4132	3214	1432	4312	276	672
1234	3412	4132	3214	1432	4231	3421	3142	4312	276	672
1234	3412	1432	3142	4312	4213	1342	3241	4132	300	1500
1234	3412	4213	1342	3142	4312	1432	3241	4132	300	1500
1234	3412	4321	4132	3241	4312	1342	4231	3142	308	576
1234	3412	4231	3142	4321	4132	3241	4312	1342	308	576
1234	3412	1342	4321	4132	3241	4312	4231	3142	308	576
1234	3412	4231	3142	1342	4321	4132	3241	4312	308	576
1234	3421	3412	4231	3142	4312	1432	3241	4132	308	576
1234	3421	3142	4231	3412	4312	1432	3241	4132	308	576
1234	3421	3142	4312	4231	3412	1432	3241	4132	308	576
1234	3412	4231	3421	3142	4312	1432	3241	4132	308	576
1234	3412	4213	1342	4132	3142	3214	1432	4312	308	1440
1234	3412	4132	3142	3214	1432	4213	1342	4312	308	1440
1234	3412	4213	1342	3214	1432	3142	4132	4312	308	1440
1234	3412	4213	1342	3214	1432	4132	3142	4312	308	1440
1234	3412	4132	3142	4213	1342	3214	1432	4312	308	1440
1234	3412	4132	3214	1432	4213	1342	3142	4312	308	1440
1234	3412	4213	1342	3142	4132	3214	1432	4312	308	1440
1234	3412	4132	4213	1342	3214	1432	3142	4312	308	1440
1234	3421	3412	4231	3142	4321	4132	3241	4312	404	0
1234	3421	3142	4231	3412	4321	4132	3241	4312	404	0
1234	3412	4231	3421	4132	3241	4321	3142	4312	404	0
1234	3412	4231	3421	3142	4321	4132	3241	4312	404	0
1234	3412	4321	3142	4231	3421	4132	3241	4312	404	0
1234	3412	4231	3142	4321	3421	4132	3241	4312	404	0

Type IIIc

t = 3 (no solution)

t = 4 (Type II, index 2)

1234	2314	2134	1324	3214	3124	(6)	TR
1234	2314	3124	1324	3214	2134	(6)	TR
1234	2143	2134	1324	3142	3124	(6)	
1234	2314	2143	2134	1324	1243	(6)	
1234	1324	1243	3124	2314	2143	(6)	

1234 2314 2134 1324 1243 2134 (12)

1234 2314 3124 2134 1324 1243 (12)

t = 5 (Type II, index 3)

12345 32415 23145 31435 14235 24315 -
34125 31245 13425 43215 42135 41325 (24) TR

CHAPTER 7

CONSTRUCTING ONE DESIGN FROM ANOTHER

7.1 Summary

Methods for deriving one design from another are described. In §7.2 various methods used elsewhere in this thesis for manipulating designs are classified and described in more detail.

In §7.3 randomization procedures are discussed; although randomization has a distinct role from the methods of §7.2 the practical procedures are closely related. The close relationship between restricted randomization described by Grundy and Healy (1950) and the design key procedure is demonstrated in §7.4.

7.2 Deriving one design from another

There are numerous ways of deriving an experimental design from one or more others e.g. collapsing factor levels; we concentrate on a few which are likely to prove the most useful. Most methods reported can be used in conjunction with each other to give a primitive calculus of designs. (See Appendix A7.1 for an example.)

Because of the inherent duality, many of the operations described can be applied to both treatments and units. The duality is not complete, however, for we allow only one treatment to be associated with any unit but allow, in multiple replicate designs, several units to be associated with one treatment. Therefore some dual operations which are useful in single replicate designs may not be defined in multiple replicate designs. Also, because randomization applies to units and treatments differently some operations and their duals may not have equal usefulness.

An experimental plan can be considered as a function $f:U \to T$ defining for each unit the treatment to be applied. The function f has three stages

$$f_1 : U \to U'$$

$$f_2 : U' \to T'$$

$$f_3 : T' \to T$$

where U' and T' denote unit and treatment labels respectively. (f_2 corresponds to the design, f_1 and f_3 to unit and treatment randomization respectively.) An experimental plan is thus altered if either spaces U and T are changed or function f is changed; the latter happens if either spaces U' and T' are changed or if functions f_1 , f_2 , f_3 are changed. Spaces U and T are determined by

i) the structure formulae

ii) the levels of factors specified in the structure formulae.

The spaces can therefore be changed by

a) addition *or* removal of factors

b) altering levels of factors

c) changing operators or reordering the formulae.

Spaces U' and T' are determined by the effective block/treatment structures (§1.3) and are therefore affected by the above changes a) and b) to the block/treatment structures. Each of the functions (f_1 , f_2 and f_3) are affected by changes to either of the two spaces with which they are associated. (Functions f_1 and f_3 can have non-random components some of which are discussed in this section; randomization is discussed in §7.3.).

Many constructions involve manipulating two designs (or plans) to form a third, (e.g. Example 1.4 §1.1.1). In these, the block/

treatment structures of basic designs A_1 and A_2, say, are manipulated to yield those of final design B. The distinction between combining two designs to form a third and updating one design to form a second is not always clear for if designs A_1 and A_2 are regarded as components of a single design A then $f(A_1, A_2)$ can be regarded also as f(A). In some of the constructions described below as involving two basic designs, the more common construction involves only one design but the present form sometimes has advantages.

Example 7.1 When the complement operation is represented as the difference of two designs, the relationship between the three block structures is:

$$(b/k) - (b/k_1) \rightarrow b/(k - k_1) \quad .$$

This operation is clearly in the same category as one for subtracting blocks of one design from another:

$$(b/k) - (b_1/k) \rightarrow (b - b_1)/k \quad .$$

One general operation is therefore required instead of two.

7.2.1 Operations on individual factors

i) Permute levels this operation neither affects the structure formulae nor the effective structure formulae and is normally allowed by randomization procedures. Some control of the permutation is sometimes required; e.g. it may be used to select a particular fraction of a p^{n-k} design or to ensure that control treatments are associated with particular formal levels. Particular permutations include addition (modulo t_i) of a constant to each level of the ith factor or multiplication (modulo t_i) by a constant which is coprime to t_i .

ii) Renumber levels This operation is similar to i) but allows

that some levels may not be common to the initial and derived designs. This operation is probably most useful for 'packing' levels after some have been deleted e.g. renumbering treatments 0, 1, 4, 5, ... as 0, 1, 2, 3, ...

iii) Delete levels This operation does not affect the structure formula (unless one factor is eliminated) but does affect the numbers of levels and the total number of units. The operation is closely related to the restriction operation (§1.2) and can be used to derive asymmetric from symmetric factorial designs or ϕ series designs from lattice designs (§2.6.2). Sometimes deleting levels of one factor affects another, thus in block section, deletion of all treatment levels associated with a selected block also removes that block.

iv) Collapse levels In this operation the levels of certain factors are mapped onto a subset of levels. As a single factor operation it applies only to treatment factors because when applied to unit factors a 'replication' factor needs to be introduced to distinguish units. Use of the operation is largely restricted to some types of orthogonal main-effect designs.

7.2.2 Modifying structure formulae

The operations defined in this section do not necessarily affect the combinatorial arrangement of an experimental plan or design but do affect the labelling and can affect the effective structures and randomization. The most useful operations in structure formulae (§1.3) are cross (*), nest (/), dummy (@), product (;) and merge (:). Apart from merge these operations may be interchanged in a block structure formula without affecting the total number of units but the randomization and the strata of the associated analysis of

variance are affected. Thus e.g. dummy causes a factor to be omitted from the randomization and product causes two factors to be treated as one. For treatment structure formulae the dominant affect of these operations is on the representation of treatments.

Example 7.2

i) Interchanging rows and columns of a Latin square yields the conjugate square.

ii) Fisher and Yates (1963) give a generalized cyclic solution for a BIB (31,10, 10, 31; 3) design equivalent to the following (§2.5.1):

(00 05 11 14 22 23 07 32 34 35) mod (3,7)

(00 01 03 10 11 13 20 21 23 36) mod (-,7)

(00 01 02 03 04 05 06 07 17 27) mod (3,-)

This construction may be achieved by replacing the treatment factor T with 31 levels by the treatment (pseudo-) structure $(A_1:I_1);(B:I_2)$ where A has three levels, B seven levels, I_1 and I_2 one level (3 and 7 respectively). (Treatment I_1I_2=37 is unassigned.)

7.2.3 Interchanging block and treatment factors

Interchanging factors between block and treatment structures gives a new design if each unit of the derived design is uniquely defined. The operation has, however, proved most useful only where subsets of factors with equal levels have been interchanged, or when in a one-one definition all block and treatment factors have been interchanged.

Example 7.3

i) Interchanging treatment and column factors of a Latin square yields an adjugate design.

ii) Let design A be an incomplete block design and B be a design derived from it by interchanging block and treatment factors:

design	block structure	treatment structure
A	B_1/B_2	T@Z
B	T/Z	$B_1@B_2$

where Z in the first structure and B_2 in the second are dummy replication factors then B is the dual design of A .

7.2.4 Designs formed from two others

In these constructions design C is formed by operating on designs A and B .

Merge The n_1 units of design A are added to the n_2 units of design B to yield design C with n_1+n_2 units. The block structures differ only in the ith factor and block factor C_i is obtained by merging factors A_i and B_i :

design	block structure	treatment factors
A	$E_1 \circ A_i \circ E_2$	$\{T_1\}\{T_2\}$
B	$E_1 \circ B_i \circ E_2$	$\{T_1\}\{T_3\}$
C	$E_1 \circ (A_i:B_i) \circ E_2$	$\{T_1\}\{T_2\}\{T_3\}$

where E_1 and E_2 are substructures (either may be absent) and $\circ$ denotes an operator; sets $\{T_i\}$ are disjoint sets of factors, some possibly empty. Factors not present in the treatment structures of designs A and B may be added as one-level dummy factors. A_i or B_i can be one-level dummy factors; $(A_i:B_i)$ can be replaced by factor C_i giving block structure $E_1 \circ C_i \circ E_2$.

Example 7.4

i) Addition of a superblock to a resolvable design:

design	block structure	treatment structure
A	SBLOCK/BLOCK/PLOT	T
B	I/BLOCK/PLOT	T
C	(SBLOCK:I)/BLOCK/PLOT	T

ii) Forming a block design as a mixture of factorials:

design	block structure	treatment structure
A	BLOCK/PLOTA	F_1
B	BLOCK/PLOTB	F_2
C	BLCOK/(PLOTA:PLOTB)	$F_1 : F_2$

where F_1 and F_2 denote structure formulae.

Remove Design B is contained in design A: the units of B are deleted to leave C (see Example 7.1). The operation is essentially the reverse of the merge operation but is less useful and can usually be achieved by deletion of factor levels (cf. block section §2.4.1).

Join Designs A and B with identical block structure are joined to form design C by superimposing the treatments of design B on those of design A. If A and B contain a common treatment factor T then the level in design C is the sum (modulo t) of the levels in A and B. The operation can be used to superimpose treatments on an existing design (no common treatment factors) or to change treatments (identical treatment structures).

design	block structure	treatment factors
A	E	$\{T_1\}\{T_2\}$
B	E	$\{T_1\}\{T_3\}$
C	E	$\{T_1\}\{T_2\}\{T_3\}$

Product The product of design A with n_1 units and design B with n_2 units is design C with $n_1 n_2$ units. The treatment allocated to unit $n_2 i+j$ of C is the join (see above) of treatments allocated to units i and j of designs A and B respectively.

A	E_1	$\{T_1\}\{T_2\}$
B	E_2	$\{T_1\}\{T_3\}$
C	$E_1 \circ E_2$	$\{T_1\}\{T_2\}\{T_3\}$

where E_1 and E_2 are structure formulae, o is an operator and $\{T_i\}$ are disjoint sets of treatment factors.

Example 7.5

i) Forming a split-plot design from a randomized block design:

design	block structure	treatment structure
A	BLOCK/PLOTA	F_1
B	PLOTB	F_2
C	BLOCK/PLOTA/PLOTB	$F_1 * F_2$

ii) Forming an incomplete block design as the direct product of two others:

A	B_1/B_2	T_1
B	B_3/B_4	T_2
C	$(B_1;B_3)/(B_2;B_4)$	$T_1 ; T_2$

Compound: Let A be a design with a block factor K with k levels and B be an incomplete block design with a treatment factor K' with k levels. Replace factor K in A by the block structure of design B such that when level i of K occurs it is replaced by the units of B containing level i of K' . (See §2.5.2).

design	block structure	treatment factors
A	E_1 o K o E_2	$\{T_1\}$
B	E_3	K'
C	E_1 o E_3 o E_2	$\{T_1\}$

Example 7.6 A is a fundamental design for 12 treatments in four groups of three units and B is a BIB (4, 3, 2, 6; 1) design; the four-level block factor of design A is to be replaced:

Design A			Design B		
BlockA	PlotA	Treatment	BlockB	PlotB	Treatment
1	1	1	1	1	1
1	2	2	1	2	2
1	3	3	2	1	2
2	1	4	2	2	3
2	2	5	3	1	3
2	3	6	3	2	1
3	1	7	4	1	1
3	2	8	4	2	4
3	3	9	5	1	2
4	1	10	5	2	4
4	2	11	6	1	3
4	3	12	6	2	4

Consider treatment 3 of design B; it occurs with units (2,2), (3,1) and (6,1). In design A therefore unit 3 1 is replaced by three units:

2	2	1	7
3	1	1	7
6	1	1	7

Replacing the second and third factors by a six-level factor gives a design for 12 treatments in six blocks of six units. (It is a singular group divisible PBIB design.)

This method corresponds to methods A and B of §2.5.2. It is used by Shrikande and Raghavarao (1963) to give a recursive construction for BIB designs, by Patterson and Lucas (1962) for change-over designs and Cotter (1978) for fractional factorial designs.

The method may be extended so that two block factors in A may be replaced through a design B for two treatment factors (one of which may be a dummy replication factor.)

Example 7.7 Let A and B be the designs:

A.					B.				
	1	2	3	4		1	2	3	4
	5	6	7	8		2	3	4	1
	9	10	11	12		3	4	1	2

where in both designs columns form blocks and the rows in design B determine a (treatment) dummy replication factor. Then both block factors in A may be replaced and C (with columns forming blocks) is

1	2	3	4
6	7	8	5
11	12	9	10

If A and C are merged the resulting design is a simple rectangular lattice. This compound-plus-merge operation corresponds to the expansion operation of Williams, Patterson and John (1977) (§2.6.4).

7.3 Randomization procedures

In §§7.3 and 7.4 we describe some practical randomization procedures which could be included in a small computer program. In §7.3.1 we comment on the requirements for pseudo-random number generators and random permutation routines. In §7.3.2 we show how for designs with simple block structure the nesting matrix M (§1.3) can be used to determine the randomization applicable to each factor in the block structure. In §7.3.3 we comment briefly on two forms of randomization which though not widely used could be considered for inclusion in the program. Restricted randomization is described in §7.4. The use of randomization to determine treatment allocation in sequential clinical trials (Pocock and Simon (1975)) is not discussed.

There has been extensive debate on the need for randomization in experiments (see e.g. Harville (1975a), Kempthorne (1977)). We avoid involvement in this debate. Fisher (1935) established randomization as a fundamental step in constructing an experimental plan, a position still recognised by most research workers in agricultural experimentation. An experimental plan program intended for these research workers which does not contain randomization procedures is therefore

unacceptable. Though workers in industrial experimentation are more divided on the issue there is sufficient support (e.g. Box, Hunter and Hunter (1978)) to make the inclusion of randomization facilities important. However, it is essential that the program user should be able to choose whether or not to randomize and to determine the form of randomization he requires.

7.3.1 Pseudo-random numbers and permutations

Unlike simulation studies or randomization tests an experimental plan program needs to produce only a few random numbers; rarely are more than one thousand numbers required. The randomization procedures can therefore be quite slow but they should be sound.

Knuth (1969, chapter 3) discusses various linear congruential generators of the form

$$x_n = (ax_{n-1} + c) \bmod m \qquad n \geq 1$$

$$y_n = x_n/m$$

where a, c, m and x_n are integers, $0 \leq x_n < m$, and $0 \leq y_n < 1$. He summarizes his discussion (p.155) as follows:

i) x_o may be chosen arbitrarily; the current date and time should suffice.

ii) m should be large and x_n determined exactly.

iii) a should be in the range $\sqrt{m} < a < m - \sqrt{m}$ (preferably $a > m/100$).

iv) If m is a power of 2 then a should not have a simple binary representation and should satisfy $a \bmod 8 = 5$; c should be odd.

v) decisions based on x_n should be influenced by the most significant digits.

He subjects several generators to an efficient spectral test and finds values of $a = 5^{15}$, 3141592221 and $2^{23} + 2^{14} + 5$ with $m = 2^{35}$ satisfactory generators. (The last of these with $c = 1$ is used

as the standard generator in GLIM.)

The standard procedure for generating a random permutation of n integers is to select one number at random from the list, delete it, select another from the $n-1$ remaining numbers and so on. It corresponds to expressing any number in the range $0 \leq k < n!$ in the form

$$k = a_1(n-1)! + a_2(n-2)! + \ldots + a_{n-1}.1$$

where $0 \leq a_i \leq n-i$, and selecting the a_i and hence k at random. (If n is sufficiently small k can be selected directly and the a_i derived from k.) This method though inconvenient for hand use is well-suited to computer applications and the advantages of the two stage procedure described by Rao (1961b) and reported by Fisher and Yates (1963) appear to be negligible. Plackett (1966) compares the efficiency of various permutation procedures.

7.3.2 Determining the randomization from the block structure

Block structure formulae determine both the default randomization for units and the strata in the null analysis of variance. Routines for interpreting structure formulae in data analysis programs may therefore be used to control randomization. In this section the use of such routines for this purpose is described.

For completely nested structures there is one stratum for each block factor and the randomization and formulae expansion are obvious. For structures which include crossing the numbers of factors and strata differ but the randomization to be applied can be determined from a subset of the strata.

Rules for expanding structure formulae to identify the strata are readily derived:

let F_1 on F_2 be block structure formulae containing factors

$X_1, X_1 \ldots X_n$ and $Y_1, Y_2 \ldots Y_m$ respectively then the following expansions hold

$$F_1/F_2 = F_1 + fac(F_1).F_2$$

$$F_1 * F_2 = F_1 + F_2 + F_1.F_2$$

where $fac(F_1) = X_1.X_2.X_3 \ldots X_n$.

Example 7.8 The strata for a Latin square design with split plots is given by

$$(A*B)/C = A + B + A.B + A.B.C \ .$$

The following theorem shows that the same expansions can be used to define the default randomization procedure for designs with simple block structure. (Firstly note that only * and / may occur in the block structure and no factor may occur twice.)

Theorem 7.1 The randomization applied to any factor B_i is determined by the smallest term containing B_i in the expanded formula; levels of B_i are randomized within each combination of levels of all factors in that term.

Proof Assume the theorem holds for all factors in formulae F_1 and F_2. The theorem then holds in the following cases:

i) any factor B_i in F_1 of formula F_1/F_2 because neither the randomization of B_i nor the smallest term containing B_i change;

ii) any factor B_i in F_2 of formula F_1/F_2 because B_i is nested in the factors determined by expanding F_2 plus all factors in F_1 and the smallest term containing B_i in the expansion of F_1/F_2 is $fac(F_1)$. {smallest term in F_2 containing B_i} .

iii) any factor B_i in F_1 or F_2 of $F_1 * F_2$ because randomization of all factors is unchanged by the crossing operation and the smallest terms are also unchanged.

In all three cases the smallest term containing B_i is unique in the combined expression if it is unique in the component

expressions. The theorem holds for $n = 2$ factors in the expressions B_1/B_2 and B_1*B_2 and the proof follows by induction.

Let $B_1, B_2, \ldots, B_m$ be block factors occurring in that order in the block structure then the matrix M constructed as follows

$m_{ij} = 1$ if factor j occurs in the smallest term containing factor i.

is the nesting matrix defined in §1.3. This matrix can therefore be used to control the randomization process. The nesting matrix can be readily derived from the expansion obtained from the formula interpretation routines given by Rogers (1973). Alternatively simpler routines may be used to derive the nesting matrix (§10.6.5) and the expansion derived from the nesting matrix. Nesting matrices therefore supply an alternative form of specifying randomization but not all matrices correspond to structure formulae.

Various forms of permuting factor levels should be available. For simple block designs standard randomization or restricted randomization (Grundy and Healy (1950)) may be required. For other designs the permutation of levels for some factors may be severely restricted or totally suppressed (e.g. the 'period' factor in change-over designs.) The treatment structure can sometimes be used to control randomization of treatments in a similar manner; in a variety trial with a second treatment factor, for example, the levels of the variety factor can be randomized while the levels of the second factor are fixed.

7.3.3 Constrained and weighted randomization

Sutter, Zyskind and Kempthorne (1963) consider constrained randomization for t treatments in blocks of kt units where the permutation of units in each block is selected at random from a

subset P of the possible permutations. They require that all treatments are equally likely to be allocated to any unit and that in the absence of treatment effects the expectations of the treatment and error mean squares are equal. There is a correspondence between the subset P and a resolvable BIB (kt, n, k, nt; λ) design where n is the number of permutations in P (and can be > 15) and $\lambda = n(k-1)/(kt-1)$; each superblock of the BIB design corresponds to a permutation in P and the treatments in ith block of the selected superblock determine to which units the ith treatment is allocated. This procedure may be readily included in a computer program because it involves three standard steps:

i) construct a suitable resolvable BIB design

ii) select a superblock at random, randomise the blocks.

iii) form the dual design of this superblock.

Randomization analysis for simple block designs in the presence of a covariate is biased. Cox (1956) shows that for completely randomized design, weighting the randomization by S_z the residual mean square on the analysis of variance of the covariate yields an unbiased randomization analysis. (The weighting also has the advantage of increasing the probability of choosing allocations such that the treatment means for the covariate are nearly equal i.e. of choosing more efficient allocations.) The weighting may be determined by first deriving the design having optimal allocation of treatments with residual mean square S_{max} and then accepting any unweighted randomization with probability S_z/S_{max}. Harville (1975b) presents an algorithm for the optimum allocation of units; should this routine be included in the experimental plan program then weighted randomization becomes a trivial extension.

7.4 Restricted randomization

Grundy and Healy (1950) give a method for obtaining a restricted randomization for blocks of eight or nine plots with special reference to p^n factorial designs in quasi-Latin squares. In this section we give an alternative presentation of restricted randomization which is equivalent to that of Grundy and Healy as long as the construction of the design (i.e. allocation of treatment labels to unit labels) meets some basic conditions.

We illustrate the proposed procedure with an example for blocks of eight units. The following set of plots effects (§3.3) α, β, $\alpha\beta$, ... have the property that no effect has like sign on four successive units and only one effect has the sequences + + + and - - -

$$\begin{aligned}
\alpha &: + + + - + - - - \\
\beta &: + + - - - + + - \\
\alpha\beta &: + + - + - - - + \\
\gamma &: + - - - + + - + \\
\alpha\gamma &: + - - + + - + - \\
\beta\gamma &: + - + + - + - - \\
\alpha\beta\gamma &: + - + - - - + +
\end{aligned} \qquad 1)$$

Restricted randomization is achieved by allocating three linearly independent effects from this set at random to plot effects P_1, P_2 and P_3. If the design is constructed by identifying treatment effects with plot effects P_1, P_2 and P_3 restricted randomization is achieved.

The set of effects 1) can be generated from the α, β and γ effects. Replacing + by 0 and - by 1 and working additively modulo 2 we get the alternative representation

$$\begin{array}{lcccccccc} \alpha: & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ \beta: & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ \gamma: & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{array} \qquad 2)$$

This array, which we denote U, is complete in the terminology of Patterson (1976) (see §2.3) and remains complete after interchange of symbols 0, 1 for any row. Choose three linearly independent effects at random $\alpha\gamma$, $\beta\gamma$ and β, say, and identify with plot factors P_1, P_2 and P_3

$$\begin{array}{lcccccccc} P_1: & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ P_2: & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ P_3: & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array} \qquad 3)$$

This array E is derivable from array U by matrix multiplication

$$E = L^T U \qquad 4)$$

where

$$L^T = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad 5)$$

The randomization procedure thus far corresponds to selecting matrix L^T at random from the class of non-singular $k \times k$ matrices ($k=3$) with elements modulo $p(=2)$. The randomization is completed by randomly permuting the levels of each plot factor.

The procedure we propose for restricted randomization is then

1) Select and store complete arrays D which give rise to plot effects with desirable properties. (It is not essential to have more than one array.)
2) Select an array D at random.
3) For each row of D randomly permute the levels 0, 1, ..., p-1, to form U.

4) Construct a random non-singular matrix L^T and form a new complete array $E = L^T U$ containing the plot factor levels.

For the design key method plot effects are identified with treatment effects. In the randomization plot effects are identified with units effects and so, when the two procedures are used in conjunction, treatment effects can be identified with unit effects and treatments derived directly from U

$$T = K_1^T L^T U \qquad 6)$$

where K_1 is the appropriate sub-matrix of the design key matrix.

Notes

i) Only one array D is necessary to ensure the validity of the procedure, but selection of D at random from a suitable set causes no harm.

ii) the plot effects given by Grundy and Healy (1950) are suitable for blocks of dimension 8×1 and 9×1 and those given by Dyke (1964) are suitable for blocks of dimension 16×1 . These are not necessarily appropriate for blocks with different dimensions and other schemes may be preferable.

Appendix A7.1

Example of construction techniques for a BIB design

A BIB (25, 9, 9, 25, 3) design D is constructed using a variety of techniques. (The example is chosen for demonstration; it is recommended (§2.5.1) that the design given by Fisher and Yates (1963) be stored complete.)

Let R, P and T, respectively, denote the block, plot and treatment factors for design D. Let

$$R = R_1 : R_2 \qquad P = P_1 : P_2$$

where R_1, R_2, P_1, P_2 have 18, 7, 6, and 3 levels respectively. Represent the 25 treatments as a subset of levels from a 36-level factor T'. Replace T' by pseudo-factors T_1 and T_2 with 9 and 4 levels respectively i.e. $T' = (T_1 ; T_2)$; only one treatment corresponds to levels 8 or 3 of T_1 and T_2. Let

$$T_1 = T_3 : T_4$$

or

$$T_1 = T_5 : T_6$$

where T_3, T_4, T_5, T_6 have 6, 3, 7 and 2 levels respectively then the design may be constructed by merging three designs A, B and C with structures

A	R_1/P_1	$T_3;T_2$
B	R_1/P_2	$T_4;T_2$
C	R_2/P	$T_5;T_2$

Merging is the sequence: merge A, B to form D'; merge D', C to form D where D' has block structure:

$$D: \; R_1/P \; .$$

Design A: Generate 18 blocks from initial blocks:

(00 10 20 30 40 50) (00 11 20 31 42 52) (00 12 22 31 40 51)

(00 10 21 32 42 51) (00 12 21 30 41 52) (00 11 22 32 41 50)

mod (1,3)

Design B: Generate six blocks from initial blocks:

(60 70 71) (60 70 83) mod (1,3) .

Repeat each block three times (cf. the product operation) and merge A and B to form D' so that the three blocks in any superblock of A have the same block of B appended.

Design C: Generate seven blocks from initial blocks:

(00 01 02 10 11 12 30 31 32) mod (7,1)

or form the equivalent singular PBIB (2) design by compounding the fundamental group divisible design with a cyclic BIB (7, 3, 3, 7; 1) design. Merge D and D'.

CHAPTER 8

OBTAINING EFFICIENT DESIGNS

8.1 Summary

Several topics related to the search for efficient designs and their evaluation are discussed. In §8.2 measures of design efficiency which are in common use are described. One measure, the harmonic mean efficiency factor $\bar{E}$ is particularly important in equireplicate block designs and in §8.3 a brief review of the literature on upper bounds for $\bar{E}$ is given. In associated appendices (A8.1 - A8.3) further results on $\bar{E}$ and upper bounds are presented; these results relate the efficiency of a design to the efficiency of one derived from it by some of the methods of Chapter 7. In Appendix A8.4 some upper bounds for the harmonic mean efficiency factor of a resolvable row and column design are presented.

Detailed study of optimal design procedures does not fall within the scope of this thesis. However, efficient designs are required and in §8.4 we review optimal design procedures which could be included in the computer program being considered. In §8.5 we discuss some practical problems associated with including a design evaluation procedure in the program.

8.2 Measures of efficiency

Programs for constructing experimental plans can benefit from inclusion of procedures for comparing rival designs; in this section we provide a brief description of the approach most commonly used when the response variable y is expected to be related to independent or controllable variables $\underline{x}$ by the model $E(y) = f(\underline{x},\underline{\theta})$ where the form of f is specified and $\underline{\theta}$ is a vector of unknown

parameters to be estimated. For response surface designs the chief interest is normally in obtaining good estimates of the surface and often the overall measure of efficiency is related to the maximum variance of any predicted value in the design space (i.e. the space defined by the bounds placed upon the independent variables). For regression and block designs the interest centres on the parameters $\underline{\theta}$ and overall measures of efficiency are based on average variances of contrasts among the θ_i. At the time of design construction, these variances are unknown, and two designs may only be compared if some assumption is made about the respective error variances. Usually the error variances can be regarded as the same for each design but some other assumption may be needed for designs with different block sizes.

For brevity we restrict attention to block designs and the linear model likely to be of greatest use in the computer program we are considering:

$$\underline{y} = X_1\underline{\theta}_1 + X_2\underline{\theta}_2 + \underline{\varepsilon} \qquad E(\underline{\varepsilon}) = 0, \quad Var(\underline{\varepsilon}) = \sigma^2 I \quad . \qquad 1)$$

where $\underline{\theta}_1$ and $\underline{\theta}_2$ are $p \times 1$ and $v \times 1$ vectors of unknown parameters; X_1 and X_2 are matrices of independent variables (possibly functions of yet others); good comparisons among the $\hat{\theta}_2$ are required. This model is used in constructing optimal block designs (Freeman (1976b), Jones (1976)), change-over designs (Hedayat and Asfarinejad (1978)) and designs with covariates (Harville (1975b)).

The estimates of θ_2 are given by

$$\hat{\underline{\theta}}_2 = (X_2^T R_1 X_2)^- X_2^T R_1 \underline{y} = C^- X_2^T R_1 \underline{y} \qquad 2)$$

where $R_1 = I - X_1(X_1^T X_1)^- X_1^T$, the residual operator for parameters $\underline{\theta}_1$ and C^- is the Moore-Penrose inverse of the matrix of reduced normal equations C. If $L^T\underline{\theta}_2$ is an estimable function of the θ_{2i}

and L has normalized coefficients so that $L^T L = 1$ then

$$\sigma^2 \mu_{min} \leq \text{var}(L^T \hat{\theta}_2) = L^T C^- L \leq \sigma^2 \mu_{max} \qquad 3)$$

where μ_{min} and μ_{max} are the smallest and largest non-zero latent roots of C^-. Pearce, Calinski and Marshall (1974) call the set of contrasts $P\theta_2$, derived from the latent vectors of C^-, the basic contrasts. The contrasts have covariance matrix

$$\sigma^2 P C^- P^T = M\sigma^2$$

where M is a diagonal matrix of latent roots of C^-. (For block designs the zero root of C corresponding to the mean is omitted and M has dimension $v-1$.) Measures of efficiency may thus be based on the latent roots μ_i of C^- or on the latent roots λ_i of C. The most widely used measures are

$$\begin{aligned} &\text{A.} && \text{tr } M = \Sigma \mu_i && = \Sigma \lambda_i^{-1} \\ &\text{D.} && \det M = \prod_i \mu_i && = \prod_i \lambda_i^{-1} \\ &\text{E.} && \mu_{max} && = \lambda_{min}^{-1} \quad . \end{aligned} \qquad 4)$$

Optimal designs are those for which one (or more) of these criteria are minimized. They are known as the A, D or E-optimal designs respectively. The D-optimality criterion has been most widely used by workers on regression designs and the A-optimality criterion by workers on block designs. All are mean criteria, however, and have some disadvantages and designs which do well on one but badly on others should be treated with caution.

Kiefer (1975) shows that for block designs a useful family of criteria is

$$\left(\sum_i \lambda_i^{-p}\right)^{1/p} \qquad (p \geq 0) \qquad 5)$$

where summation is over the non-zero roots of a connected block design. The values $p = 1$, 0 and ∞ then yield the criteria 4). A design with $C = \alpha I + \beta J$ (where J has all elements unity) is optimal for all members of this family;

C has this form for, among others, BIB designs. This result shows that where efficiency has been obtained by striving for balance the resulting designs probably have good all-round properties. For binary proper equireplicate block designs the off-diagonal elements of C are determined by the concurrence matrix. A useful restriction therefore when looking for efficient designs is to consider those designs for which the range (or variance) of the off-diagonal elements in the concurrence matrix is minimized. Designs for which the off-diagonal elements differ by at most one are called regular graph designs by John and Mitchell (1977). These authors, John, Wolock and David (1972) and Williams, Patterson and John (1977) all use regular graph designs in obtaining optimal or efficient block designs. Because of their all-round properties these designs should prove to be robust.

Commonly, optimality or efficiency in block designs is assessed not on the matrices C and M but on very closely related matrices. The most standard substitution is

$$A = (X_2^T X_2)^{-\frac{1}{2}} C (X_2^T X_2)^{-\frac{1}{2}}$$

$$= I - (X_2^T X_2)^{-\frac{1}{2}} X_2^T T_1 X_2 (X_2^T X_2)^{-\frac{1}{2}}$$

where $T_1 = I - R_1$ and A is the standardized matrix (§2.2.1). For equireplicate designs the roots are λ_i/r and yield a measure of the efficiencies of the basic contrasts relative to a randomized block design with the same replication and error variance. Omitting the root corresponding to the mean the remaining roots are the efficiency factors (§2.2.1). The harmonic mean efficiency factor is equal to the mean efficiency factor of all pairwise comparisons among the $\underline{\theta}_2$. (For designs with unequal replication interpretation is less direct and the choice of using roots of matrix A or C

remains open (see, e.g. Pearce (1970)). Freeman (1976) and Jones (1976) suggest as an optimality criterion the weighted sum of the variances of selected orthogonal contrasts. This is equivalent to minimising

$$\begin{aligned} & \mathrm{tr}(w^{\delta/2}\ B^T C^- B\ w^{\delta/2}) \\ = \ & \mathrm{tr}(Bw^{\delta}\ B^T\ C^-) \end{aligned}$$

where the contrasts are held in the columns of B and $w^{\delta} = \mathrm{diag}(w_1, w_2 \ldots w_k)$ is the weighting matrix.

Note: The work of John and co-workers on efficient regular graph designs and Jones (1976) can be compared with the work on symmetric p^n factorials presented in §3.4 and 3.5. The regular graph approach corresponds closely to that of §3.4 where contrasts have equal importance and standard designs are wanted. The approach of Jones corresponds to that of §3.5 where more is known about the contrasts of interest and tailor-made designs are required; efficient designs may then be specific to the requirements and should be treated with caution if very unbalanced.

8.3 Upper bounds for the harmonic mean efficiency factor of block designs

The arithmetic mean of the efficiency factors is readily derived from the trace of matrix A (§8.2) providing an upper bound E_0 for the harmonic mean efficiency factor $\bar{E}$, the bound being achieved for balanced designs. In many instances the true upper bound is much smaller than E_0 and more realistic bounds are required. Computation of such bounds could be built into a construction program for comparison with the $\bar{E}$ values achieved. We briefly review the literature on upperbounds.

A simple improvement on the upper bound is given by Williams, Patterson and John (1976) namely,

$$U_p = 1 - \frac{k'(v-k')}{k^2(v-1)}, \quad k' = k \bmod v \quad .$$

For binary designs, $U_p = E_o$ but for non-binary designs, $U_p \leq E_o$.

Pearce (1968) restricts attention to equireplicate designs for which the elements n_{ij} of the incidence matrix differ by no more than 1. He then adds constraints to the standardized matrix A to make it non-singular of the form $I - H$ and approximates the inverse $(I-H)^{-1}$ by $I + H + H^2$ to give an upperbound. Jarrett (1977) observes that if the sums of squares of the off-diagonal elements of the concurrence matrix is unknown then Pearce's (1968) bound is inferior to E_o . Williams and Patterson (1977) improve on Pearce's bound when the off-diagonal elements of the concurrence matrix differ by no more than one. They replace Pearce's matrix H by

$$H = aJ + Z$$

where $ZJ = 0$ and expand $(I-Z)^{-1}$ as far as Z^3 to yield the upper bound.

Conniffe and Stone (1974) present upper and lower bounds for $\bar{E}$ when the efficiency factors are not all equal. The upper bound is achieved when all roots are equal except one which exceeds the rest by an amount proportional to the variance of the off-diagonal elements of the concurrence matrix, so if the latter is known the bound can be determined. Jarrett (1977) generalizes these results. Williams and Patterson (1977) point out that for the class of designs they are considering, their bound compares favourably with that of Jarrett.

Patterson and Williams (1975) show that when $v > b$ improved upper bounds are obtained by first determining bounds for the dual design and then setting $v - b$ roots to unity. When the design is resolvable $r - 1$ roots are unity and a further improvement can be

obtained. Both techniques can be used to improve upper bounds given above.

In Appendix A8.3, we derive upper bounds for resolvable row and column designs.

8.4 Optimal designs

The theory and construction of optimal designs overlaps with various topics discussed in this thesis. Although the construction is probably best achieved by purpose-built programs, the inclusion of a routine within a small general program may be practicable. We therefore present a brief review of the literature with emphasis on standard algorithms for linear models and block designs. More extensive reviews and bibliographies are given by St. John and Draper (1975) and Ash and Hedayat (1978).

Kiefer and co-workers laid the foundations for much recent work in optimal experimental design (e.g. Kiefer (1960), Kiefer and Wolfowitz (1959)). They introduced measure designs which essentially contain r distinct points in a specified design space R with weight w_i applied to the ith distinct point where $w_i > 0$ and $\Sigma w_i = 1$. When the w_i are multiples of $1/n$ the measure design yields an exact design for n units, nw_i units being located at the ith distinct point. If the nw_i are not integer the resulting design is an approximate design for n units. Consider the model

$$y = \sum_{i=1}^{n} f_i(x)\theta_i + \varepsilon_i \qquad E(\varepsilon_i) = 0 \quad var(\varepsilon_i) = \sigma^2 I \qquad 1)$$

where y is the observed variable, x is a $p \times 1$ vector of predictor variables, and θ_i are unknown parameters to be estimated. Let the experimental region be denoted by R_n then the practical problem in optimal design studies is to find an exact design with

n points in R which is in some sense optimal for estimating the response surface $\hat{y}$ or the parameters $\underline{\theta}$. The optimality criteria most commonly used are described in §8.2.

Determination of the exact design may be aided by using an optimal measure design as an approximation from which to start the search. This approximation does not necessarily lead to a globally optimal exact design for although there are algorithms which always yield globally optimal measure designs (e.g. Fedorov (1972, p.102)) there are none for exact designs.

The measure design algorithms work essentially by starting with a non-degenerate design and then repeated steps add weight to points of maximum variance or remove weight from points of minimum variance. Algorithms for exact designs are approximations to this process; the most widely used algorithms are based on variations of two somewhat similar algorithms given by Fedorov (1972) and Wynn (1972). Both algorithms start with an arbitrary n-point design and then repeatedly exchange points in the design with points outside until an optimum is obtained. In the former, two points are exchanged at each step so as to give maximum improvement. In the latter, a point is added to the design to give maximum improvement and then a point is subtracted to give minimum deterioration. Mitchell (1974) modified the second algorithm to allow addition and subsequent subtraction of more than one point. The essential difference between the algorithms of Fedorov and Wynn is that the former should achieve the maximum in fewer but somewhat larger and more expensive steps.

Cook and Nachtsheim (1980) compare the effectiveness of five algorithms for finding a range of exact D-optimal designs. The algorithms are those of Fedorov, Wynn and Mitchell

and two others which are similar. They consider various models and design spaces and find that the methods giving the most efficient designs are the most demanding of computer time. They make recommendations but these should be treated with caution for they used the same stopping rule for all methods and their results probably reflect the way in which the algorithms approach the stopping rule. Galil and Kiefer (1980) make a few minor changes to Mitchell's (1974) DETMAX algorithm and report up to 50-fold increases in speed. It is possible that similar improvements could be obtained for the other algorithms.

The construction of optimal block designs and the optimal allocation of units require slightly different algorithms. The problem is not the selection of points in continuous space but the allocation of treatments to units. Typically the replication is fixed or determined within narrow limits and emphasis is on the interchange of treatments between two units in different blocks or with different levels of the covariate. Eccleston and Jones (1980a, b) divide the process of obtaining an optimal design into two stages, the first (exchange) stage determines the replication and the second (interchange) the allocation of treatments to blocks. They give procedures for updating the matrix C^- (§8.2) at either stage.

Methods for seeking optimal block designs are still at an early stage of development. Two areas of research that might prove profitable are

i) the use of regression updating techniques based on Given's method (Chambers (1971), Gentleman (1974)) to provide exchange and interchange algorithms and

ii) the application of simple interchange algorithms to α or ϕ-series designs where maximum efficiency has not been achieved.

Optimal design procedures should be treated with caution; there is much to be said for restricting attention to a class of desirable designs and selecting the most efficient of these. Yates (1970, p.276) gives pertinent remarks on the rival merits of factorial and response surface designs. Kennard and Stone (1969) make similar remarks and describe a procedure for simulating factorials by arranging for the points to be evenly distributed throughout the design space. This method which has been somewhat neglected in the literature would appear to have some use in agricultural trials.

8.5 Design evaluation

The decision as to whether a design construction program should also contain evaluation procedures is not straightforward and is affected by local facilities. On the one hand designs can be passed to a statistical analysis program for evaluation, and on the other hand, if the construction program caters for a restricted range of designs, then specific evaluation routines can be built into the program. For general purpose programs the provision of special evaluation routines for all possible designs is unwieldy, but the provision of none may be unsatisfactory. We consider design evaluation procedures for small design programs.

Design evaluation is distinguished from the associated analysis by the absence of data. This absence affects both the operations required and the amount of space available, e.g. procedures for the recovery of inter-block information are redundant. The evaluation routines should be balanced for complexity and demand and there need be no special provisions for large complex designs of a type rarely

required. Evaluation of many designs from a specific class could warrant a special program. There is little need for routines evaluating highly orthogonal constructions for the properties are then readily deduced without recourse to an evaluation procedure. (A recursive procedure such as described by Wilkinson (1970) is thus of somewhat limited value.)

The principal requirements of an evaluation procedure is that it should provide for each design:

a) information on the variance - covariance matrix of selected estimable contrasts,

b) the degrees of freedom for the hypothesis associated with any set of estimable contrasts and

c) various measures of efficiency.

There is a need for some basic matrix routines but sensible programming should enable matrix inversion and latent root procedures to be restricted to symmetric matrices. To determine the requirements more specifically we note in Table 8.1 the main classes of design discussed in Chapters 2-7 and comment on the form of the reduced matrix of normal equations C or the equivalent matrix for the dual design C_* .

There are three forms of C which appear to merit special routines:

i) C is a general positive semi-definite matrix;

ii) C is a symmetric block circulant matrix;

iii) C has very few distinct latent roots.

For matrices C of types i) and iii) only elements above the diagonal need be stored; for those of type ii) considerable savings in storage space may be made by employing the structure of the matrix.

Table 8.1.

Type of design	Requirements
BIB	C has form $aI + bJ$ with a single non-zero latent root;
PBIB(r)	C has r distinct latent roots. For small r special procedures can be advantageous;
cyclic	C is a symmetric circulant matrix†;
generalized cyclic	C is commonly a symmetric block circulant matrix†;
lattice designs	the designs may be treated as quasi-factorial, PBIB(2) or generalized cyclic as desired;
α-series	C_* is a symmetric block circulant matrix;
ϕ-series	C_* is commonly a symmetric block circulant matrix;
'orthogonal' p^n factorial	subsets of matrix C are sometimes required to determine partial confounding;
'optimal' p^n factorials	usually C has few distinct roots Srivastava (1978, p.301)) but general inversion procedures could be required;
asymmetric factorials	when cyclic methods are used C is commonly a symmetric block circulant matrix otherwise it can have general form;
orthogonal Latin squares	C has form $aI + bJ$
change-over, superimposed and serially balanced	when cyclic methods are used C is commonly a symmetric block circulant matrix otherwise it can have general form;
others	C has general form.

† Circulant and block circulant matrices are defined in Appendix 8.5 and some properties are noted.

In most cases listed above where C has general form the total number of parameters in the reduced normal equations is relatively small, e.g. a second order response-surface model for eight factors requires the estimation of 36 parameters. With no requirement to store data most small computers can cope with routines for finding the latent roots and generalized inverses of symmetric matrices of order less than about 40×40. The limits imposed on the size of C by the size of memory are therefore rarely critical. Designs with many parameters in the reduced normal equations can be evaluated if C is a block circulant matrix although it may be necessary to store C and the variance-covariance matrix in a compact form.

The problems of design evaluation within a small experimental plan program do not appear insurmountable as long as more than one analysis procedure is available. But the use of multiple routines inevitably complicates the control structure and there does not therefore appear to be a definite case either for or against the inclusion of an evaluation routine and in different circumstances either decision may be made.

Efficiency factors for the complement of a block design

Let D_1 be a proper binary equi-replicate block design with $C = rI - k^{-1} NN^T$ and standardized matrix $A_1 = r^{-\frac{1}{2}} C\, r^{-\frac{1}{2}}$ then the complementary design D_2 has standardized matrix

$$A_2 = (1-w)I + \frac{b-2r}{(b-r)(v-k)} J + wA_1 \qquad 1)$$

where $w = k^2/(v-k)^2$. Therefore, if λ_i is a non-zero latent root of A_1 then

$$\theta_i = (1-w) + w\lambda_i \qquad 2)$$

is a latent root of A_2 , i.e. the efficiency factors of design D_2 are related to those of design D_1 by equation 2).

Let E_1 be the mean of the efficiency factors for D_1 and $E_2 = (1-w) + wE_1$. We show that if $k < v/2$ i.e. $w < 1$ then

$$(E_2 - \bar{E}_2) \leq w\,(E_1 - \bar{E}_1)$$

where $\bar{E}_1$ and $\bar{E}_2$ are the harmonic mean efficiency factors for D_1 and D_2 i.e. $\bar{E}_2$ is always closer to its upper bound than $\bar{E}_1$. To prove this we note that multiplication of each efficiency factor by a constant w affects E and $\bar{E}$ in the same way so we simply need to show that

$$f(\underline{a}_i, s) = HM\,(s + a_i) - HM\,(a_i) - s \geq 0 \qquad 4)$$

where HM denotes harmonic mean and $s > 0$, $a_i > 0$. It is easily shown that for Σa_i = constant, $f(\underline{a}_i, s)$ is minimized when all a_i are equal but then $f(\underline{a}_i, s) = 0$. The inequality 4) therefore always holds and hence 3) also holds, equality being achieved when all efficiency factors are equal i.e. when $\bar{E}_2 - E_2 = \bar{E}_1 - E_1 = 0$.

This result shows that if D_1 is A-optimal then D_2 is also A-optimal or nearly so except possibly when $E_1 - \bar{E}_1$ large.

Efficiency factors of dual designs with orthogonal block structure

We show that the efficiency factors of a design with orthogonal block structure (Nelder 1965)) and its dual are closely related and often readily derived from each other.

Firstly we make a preliminary remark. If D is a design with orthogonal block structure and $\underline{t}$ denotes the full set of treatment parameters then the model for the design may be written (Nelder (1965))

$$E(y) = (T_1, T_2)\underline{t}$$
$$V(y) = \Sigma \xi_i C_i \qquad 1)$$

where $T_1^T T_2 = 0$ and C_i set of orthogonal idempotent matrices with $\Sigma C_i = I$. The information matrix for $T_1\hat{\underline{t}}$ in stratum i is given by

$$T_1^T C_i T_1 \qquad 2)$$

Theorem. Let D_1 and D_2 be designs with orthogonal block structure such that

$$D_1: \quad E(y) = (T_1, T_2)\underline{t}$$
$$\text{Var}(y) = \sum_i \xi_i C_i \qquad 3)$$

$$D_2: \quad E(y) = (B_1, B_2)\underline{t}$$
$$\text{Var}(y) = \sum_j \eta_j U_j \qquad 4)$$

where $T_1^T T_2 = 0$, $B_1^T B_2 = 0$, $\{C_i\}$ and $\{U_j\}$ are sets of orthogonal idempotent matrices and

$$C_1 = B_1(B_1^T B_1)^- B_1^T$$
$$U_1 = T_1(T_1^T T_1)^- T_1^T \qquad 5)$$

then the non-zero efficiency factors for estimating $T_1\underline{t}$ in stratum C_1 of design D_1 are equal to the non-zero efficiency factors for

estimating $B_1\underline{t}$ in stratum U_1 of design D_2. (More simply if A and B are strata occurring in the block and treatment structures respectively of design D_1 and in the other structure in design D_2 then the efficiency of A in B is that of B in A. Nothing is said about other strata.)

Proof. The information matrix for $T_1\underline{t}$ in stratum 1 of design D_1 is

$$T_1^T C_1 T_1$$

and the efficiency factors are the latent roots of A_1 where

$$A_1 = S\, T_1^T C_1 T_1\, S^T$$

where $S^T S = (T_1^T T_1)^-$. If $R^T R = (B_1^T B_1)^-$ then

$$A_1 = (S\, T_1^T B_1\, R^T)(R\, B_1^T T_1\, S^T) = D\, D^T \quad \text{say.} \qquad 6)$$

The efficiency factors for $B_1\underline{t}$ in stratum 1 of design D_2 are the latent roots of A_2 where

$$A_2 = R\, B_1^T U_1 B_1\, R^T$$
$$= (R\, B_1^T T_1\, S^T)(S\, T_1^T B_1\, R^T) = D^T D\,. \qquad 7)$$

But the non-zero latent roots of $D\, D^T$ are equal to those of $D^T D$ and the result follows.

Example 1 Block designs. Let designs D_1 and D_2 have structure

D_1 : Blocks B/P Treatments T @ X

D_2 : Blocks T/X Treatments B @ X

where the second term in the treatment structure is a dummy replication factor. The efficiency factors of T in stratum B of D_1 are those of B in stratum T of D_2. For these block structures the between blocks standardized matrices A have the form $I - \frac{J}{r} - A_w$ where the matrix J corresponds to the mean and A_w is the corresponding within blocks standardized matrix. The within

block harmonic mean efficiency factors of design D_1 and its dual D_2 are therefore readily derived and have the (well-known) relationship

$$\frac{v-1}{\bar{E}_1} = v - b + \frac{b-1}{\bar{E}_2}$$

where v and b are the number of treatments and blocks in D_1.

Example 2 Resolvable block designs. Let designs D_1 and D_2 have structures:

D_1 : Blocks $B_1/B_2/K$ Treatments $T_1;T_2$ @ S

D_2 : Blocks $T_1/T_2/S$ Treatments $B_1;B_2$ @ K

where the last term in the treatment structure may be a dummy factor. If B_1/B_2 and T_1/T_2 are treated as forming a single stratum then the within blocks efficiency factors are derived as in Example 1.

If both designs are resolvable then T_1 and B_1 are orthogonal to blocks in D_1 and D_2 respectively and the non-zero efficiency factors in the combined block strata are the same as those for $T_2+T_1.T_2$ or $B_2+B_1.B_2$ in the respective 'blocks in superblocks' strata. Therefore $k=\min(b_1,t_1)-1$ efficiency factors are zero i.e. in the within blocks stratum k efficiency factors have value 1.

Example 3 Row and column designs. Let D_1 and D_2 have structures:

D_1 : Blocks R*C Treatments T @ X

D_2 : Blocks T/X Treatments R*C

where X in D_1 may be a dummy factor. Efficiency factors λ_i for T in stratum R.C of D_1 give values $1-\lambda_i$ for T in combined stratum R+C. These give efficiency factors $1-\lambda_i$ for R+C in stratum T of D_2 and λ_i for R+C in the within blocks stratum of D_2. The search for efficient row and column designs is therefore related to a search for efficient block designs with two factors.

The efficiency factors of the contraction of a two-replicate resolvable row and column design.

Let A be a two-replicate row and column design with D and D* its dual and contraction designs (§2.6.4) respectively. Patterson and Williams (1975) show that if A is an efficiency factor of D* then $(1 \pm \sqrt{(1-\lambda)})/2$ are efficiency factors of D. We extend this result to row and column designs. The block and treatment structures of the three designs may be written.

Design	A	D	D*
Block structure	S/(R*C)	T/X	R*C
Treatment structure	T @ X	(R*C) @ S	R'*C'

where the term X occurring in the treatment structure of design A is a dummy replication factor, S has two-levels and R' and C' are factors with the same number of levels as R and C.

Example: Let A be the design

1	4	7	10	|	1	4	6	3
2	5	8	11	|	5	2	7	8
3	6	9	12	|	12	9	11	10

then the dual is the design

block	1	2	3	4	5	6	7	8	9	10	11	12
	111	121	131	112	122	132	113	123	133	114	124	134
	211	222	214	212	221	213	223	224	232	234	233	231

where each number triplet represents levels of S, R and C.

The contraction is

11	12	32	31
22	21	13	23
34	33	24	14

where each number pair indicates the levels of R' and C' respectively.

Each treatment of the contraction occurs once with both levels of S in the dual. We show that each efficiency factor of the contraction gives rise to two efficiency factors of the dual.

Before stating the theorem we make some preliminary remarks:

i) For design A we require the efficiency factors for T in stratum $R.C + S.R.C$ which can be derived from those for T in combined stratum $S + (R+S.R) + (C+S.C)$. Stratum S may be omitted because S and T are orthogonal therefore the efficiency factors correspond to those of $(R+S.R) + (C+S.C)$ in stratum T of the dual design D (Appendix A8.2).

ii) Denote the incidence matrices for the treatments of design A with respect to rows and columns of the two superblocks by N_{1r}, N_{2r}, N_{1c} and N_{2c} respectively, then $N_{ir}J = J$; $N_{ic}J = J$; $N_{ir}^{T}J = cJ$; $N_{ic}^{T}J = rJ$. Let M_{rc} denote the incidence matrix of the contracted design for the factor R' with respect to columns then $M_{rc} = N_{1r}^{T}N_{2c}$. Similarly we obtain $M_{rr} = N_{1r}^{T}N_{2r}$, $M_{cr} = N_{1c}^{T}N_{2r}$ and $M_{cc} = N_{1c}^{T}N_{2c}$. From the above expressions we obtain $M_{rr}J = cJ$; $M_{rc}J = cJ$, $M_{cr}J = rJ$ and $M_{cc}J = rJ$.

iii) We use in the proof the notation $\tilde{J}_r = I - J/r$ and $\tilde{J}_c = I - J/c$ where $\tilde{J}_r$ and $\tilde{J}_c$ are symmetric idempotent matrices. We denote by Z the matrix:

$$\begin{bmatrix} \tilde{J}_r & O \\ O & \tilde{J}_c \end{bmatrix}$$

iv) $M_{rr} - r^{-1}c\tilde{J}_r = \tilde{J}_r M_{rr} = M_{rr}\tilde{J}_r = \tilde{J}_r M_{rr}\tilde{J}_r$. Similar relationships hold for M_{rc}, M_{cr} and M_{cc} .

Theorem If λ,μ are efficiency factors for R' and C' respectively in the R.C stratum of the contraction then

$$(1 \pm (1-\lambda)^{\frac{1}{2}})/2 \quad \text{and} \quad (1 \pm (1-\mu)^{\frac{1}{2}})/2$$

are efficiency factors for (R + S.R) and (C + S.C) in the within block stratum of the dual.

Proof The proof is in three parts a) the standardized matrix (§8.2) for the contraction is derived, b) the standardized matrix for the dual is derived and c) the relationship between their efficiency factors is obtained.

The contraction The contracted design contains all combinations of two treatment factors R' and C' with r and c levels arranged in an r×c row and column design. Using ii) and iii) the standardized matrix for the main effects of R' and C' in the R.C stratum can be represented

$$Z - MZM^T \qquad 1)$$

where Z is defined in remark iii) and

$$M^T = \begin{bmatrix} c^{-1}M_{rr} & (rc)^{-\frac{1}{2}}M_{rc} \\ (rc)^{-\frac{1}{2}}M_{cr} & r^{-1}M_{cc} \end{bmatrix} \qquad 2)$$

The dual After straightforward, but somewhat lengthy algebra, the standardized matrix for (R + S.R) and (C + S.C) in stratum T of the dual can be shown to be

$$P = \tfrac{1}{2}\begin{bmatrix} \tilde{J}_r & O & U_{rr}^T & U_{rc}^T \\ O & \tilde{J}_c & U_{cr}^T & U_{cc}^T \\ U_{rr} & U_{cr} & \tilde{J}_r & O \\ U_{rc} & U_{cc} & O & \tilde{J}_c \end{bmatrix} \qquad 3)$$

where $U_{rr}^T = c^{-1}(M_{rr} - r^{-1}c\tilde{J})$, $U_{rc} = (rc)^{-\frac{1}{2}}(M_{rc} - \tilde{J})$ and $U_{cc}^T = r^{-1}(M_{cc} - rc^{-1}\tilde{J})$ and the first r+c rows and columns refer to

parameters of the first superblock.

The relationship The latent roots of P are given by the solutions to $|P-\lambda I| = 0$. Writing $2P-\lambda I$ in the form $\begin{pmatrix} A & B^T \\ B & A \end{pmatrix}$ then

$$|2P-\lambda I| = |A||A-B^T A^- B| \qquad 4)$$

The determinant of the first term of 4) is

$$|A| = \lambda^2(1-\lambda)^{r+c-2} = (1-\lambda)^{r+c}(\lambda/(1-\lambda))^2 \qquad 5)$$

The matrix in the second term is

$$A - B^T A^- B = A - (1-\lambda)^{-1} BB^T$$

$$= (1-\lambda)I - \begin{bmatrix} r^{-1}J & O \\ O & c^{-1}J \end{bmatrix} - (1-\lambda)^{-1} MZM^T \qquad 6)$$

The second term of 6) is affected only by the zero latent roots of $A - B^T A^- B$ and its removal causes two zero roots to become unity i.e. the determinant is multiplied by $(1-\lambda)^2/\lambda^2$. It follows that

$$|2\bar{P}-\lambda I| = (1-\lambda)^{r+c}|(1-\lambda)I - (1-\lambda)^{-1} MZM^T|$$

$$= |(1-\lambda)^2 I - MZM^T| \qquad 7)$$

where $\bar{P}$ is P adjusted for the mean vectors.

Noting λ_i are the roots of $2P$ and that the matrix $Z - MZM^T$ in 1) can also be adjusted for the mean, the result follows.

Appendix A8.4

Upper bounds for resolvable row and column designs

Upper bounds for the harmonic mean efficiency factor of resolvable row and column designs are obtained for cases $v \leq s(r+c-2)$ and $v > s(r+c-2)$.

Let D be a design with s superblocks (S) of r rows (R) and c columns (C) each superblock containing $v = rc$ treatments.

Case i) $v \leq s(r+c-2)$.

The standardized matrix A for the treatments in the (R.C + S.R.C) stratum is

$$(I - \frac{J}{rc}) - \frac{1}{sc}RR' - \frac{1}{sr}CC' + \frac{2}{rc}J \qquad 1)$$

where R and C are the row and column incidence matrices. A has trace $(r-1)(c-1)$ and an upper bound for the harmonic mean efficiency factor is given by the arithmetic mean:

$$\bar{E} \leq \frac{(r-1)(c-1)}{v-1}$$

Case ii) $v > s(r+c-2)$.

T and S are orthogonal so the efficiency factors of T in combined strata S + (R+S.R) + (C+S.C) are given by those of R+S.R and C+S.C in stratum T of the dual (Appendix A8.2). The standardized matrix A for the dual is therefore:

$$\begin{bmatrix} \tilde{J}_r \otimes I - (sc)^{-1}(R^T - r^{-1}J)(R - r^{-1}J) & s^{-1}(rc)^{-\frac{1}{2}}(R^T - r^{-1}J)(C - c^{-1}J) \\ s^{-1}(rc)^{-\frac{1}{2}}(C^T - c^{-1}J)(R - r^{-1}J) & \tilde{J}_c \otimes I - (sr)^{-1}(C^T - c^{-1}J)(C - c^{-1}J) \end{bmatrix} \qquad 2)$$

where $\tilde{J}_r = I - \frac{1}{r}J$ and $\tilde{J}_c = I - \frac{1}{c}J$ are symmetric idempotent matrices.

Representing the matrix 2) as $\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$ then S_{11} has trace $(s-1)(r-1)$ with $s(r-1)$ non-zero roots and S_{22} has trace $(s-1)(c-1)$ with $s(c-1)$ non-zero roots. Allowing for $(v-1) - s(r+c-2)$ roots of unity gives the upper bound:

$$\bar{E} \leq \left\{ 1 + \left(\frac{(v-1)(s-1)}{s(r+c-2)} \right)^{-1} \right\}^{-1} . \qquad 3)$$

Case iii) Alternative form, $v > s(r+c-2)$.

Firstly, if no pair of treatments occur in the same row or column more than once then $R^T C = J$ and $(R^T - r^{-1}J)(C - c^{-1}J) = O$. Matrix A in ii) simplifies to a block diagonal matrix and the efficiency factors of the row and column design are determined directly from the component block designs. If A is not block diagonal then

$$\mathrm{tr}(A^-) = \mathrm{tr} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^- \geq \mathrm{tr} \begin{bmatrix} A_{11}^- & O \\ O & A_{22}^- \end{bmatrix} = \mathrm{tr}(A_{11}^-) + \mathrm{tr}(A_{22}^-)$$

Now $\mathrm{tr}(A^-)$, $\mathrm{tr}(A_{11}^-)$ and $\mathrm{tr}(A_{22}^-)$ determine the sum of the non-zero efficiency factors of the three dual designs and therefore

$$\frac{1}{\bar{E}} \geq \frac{1}{\bar{E}_r} + \frac{1}{\bar{E}_c} - 1 \qquad 4)$$

where $\bar{E}_r$ and $\bar{E}_c$ are the harmonic mean efficiency factors of the component block designs.

Appendix A8.5

Block circulant matrices

Block circulant matrices are defined and some results given by Williams (1975) and John (1980) presented.

Example Let A, B and C be three designs for eight treatments in eight blocks of three units generated as follows:

A (0 1 2) mod 8

B (00 11 31) (00 10 21) mod(4, -)

C (00 10 21) mod(4, 2) .

The concurrence matrices for the first three designs are:

$$A\begin{bmatrix}3&2&1&0&0&0&1&2\\2&3&2&1&0&0&0&1\\1&2&3&2&1&0&0&0\\0&1&2&3&2&1&0&0\\0&0&1&2&3&2&1&0\\0&0&0&1&2&3&2&1\\1&0&0&0&1&2&3&2\\2&1&0&0&0&1&2&3\end{bmatrix}\quad B\begin{bmatrix}3&0&1&2&0&1&1&1\\0&3&1&0&1&2&2&0\\1&1&3&0&1&2&0&1\\2&0&0&3&1&0&1&2\\0&1&1&1&3&0&1&2\\1&2&2&0&0&3&1&0\\1&2&0&1&1&1&3&0\\1&0&1&2&2&0&0&3\end{bmatrix}\quad C\begin{bmatrix}3&0&1&1&0&2&1&1\\0&3&1&1&2&0&1&1\\1&1&3&0&1&1&0&2\\1&1&0&3&1&1&2&0\\0&2&1&1&3&0&1&1\\2&0&1&1&0&3&1&1\\1&1&0&2&1&1&3&0\\1&1&2&0&1&1&0&3\end{bmatrix}$$

Let Γ_i denote a (square) basic circulant matrix with ith diagonal = 1 (§5.2) and 0 elsewhere then

$A = \Sigma a_i \Gamma_i$ where a_i is the ith element of the first row of A .

$B = \Sigma b_i \otimes \Gamma_i$ where b_i are 2×2 matrices and Γ_i are 4×4 matrices.

$C = \Sigma c_i \otimes \Gamma_i$ where c_i are 2×2 matrices and Γ_i are 4×4 matrices.

$= \Sigma_i (\Sigma_j c_{ij} \Gamma'_j) \otimes \Gamma_i$ where Γ'_j are 2×2 matrices.

The following notation is convenient for describing the above matrix operation.

Let M^T be partitioned $(M_1^T , M_2^T \ldots M_s^T)$ where the M_i^T all have the same dimension. Let P^T be partitioned similarly then define the operation o by

$$M^T \circ P = \sum_{i=1}^{s} M_i^T \otimes P_i \qquad 1)$$

Definition: Let $\underline{s}^T = (s_1,\ldots,s_n)$ where s_j are integer and $s = \prod_j s_j$ $(j = 1,2,\ldots,n)$ then a $ts \times ts$ matrix C is a block circulant $BC(t; n)$ matrix if it can be written in the form

$$C = (\ldots((\underline{\theta}^T \circ \underline{\Gamma}_1) \circ \underline{\Gamma}_2) \circ \ldots \circ \underline{\Gamma}_n) \qquad 2)$$

where $\underline{\theta}^T$ are $t \times s_1 t$ matrices and $\underline{\Gamma}_j^T = (\underline{\Gamma}_{j1}^T, \underline{\Gamma}_{j2}^T, \ldots, \underline{\Gamma}_{js_j}^T)$ $(j = 1,\ldots,n)$ and $\underline{\Gamma}_{jk}$ is the basic $(s_j \times s_j)$ circulant with kth diagonal unity. A $BC(1; 1)$ matrix is called a circulant matrix. In the above example A is circulant, B is $BC(2; 1)$ and C is $BC(1; 2)$.

The definition 2) is recursive; we derive some results for the $BC(t; 1)$ matrices:

$$C = \underline{\theta}^T \circ \underline{\Gamma} \qquad 3)$$

where $\underline{\theta}^T = (\underline{\theta}_1^T, \underline{\theta}_2^T \ldots \underline{\theta}_s^T)$ is a $t \times st$ matrix and $\underline{\Gamma}^T = (\underline{\Gamma}_1^T, \underline{\Gamma}_2^T, \ldots \underline{\Gamma}_s^T)$ is a $s \times s^2$ matrix. The extension to the general case is straightforward.

Let W be the $s \times s$ matrix with (i,j) element $\frac{1}{s}\omega_j^i$ where ω_j is the jth s root of unity. Let $\bar{W}$ be the conjugate and define Γ^* as

$$\begin{aligned} \underline{\Gamma}^* &= (I \otimes \bar{W})\underline{\Gamma} \quad \text{then} \\ C &= \underline{\theta}^T \circ (I \otimes W)\underline{\Gamma}^* \\ &= \underline{\theta}^T (I \otimes W) \circ \underline{\Gamma}^* \\ &= \underline{\theta}^{*T} \circ \underline{\Gamma}^* \end{aligned} \qquad 4)$$

where $\theta^{*T} = \{\theta_1^*, \theta_2^* \ldots \theta_s^*\}$ and $\underline{\theta}_h^* = \Sigma \omega_h^k \underline{\theta}_k$.

Now $\underline{\Gamma}^{*T} = \{\underline{\Gamma}_1^{*T} \ldots \underline{\Gamma}_s^{*T}\}$ where $\underline{\Gamma}_h^* = \bar{\gamma}_h \gamma_h^T$ and $\gamma_h^T = s^{-1}(\omega_h^0, \omega_h^1 \ldots \omega_h^s)$ i.e. the $\underline{\Gamma}_h^*$ are mutually orthogonal idempotent matrices,

and C has spectral form

$$C = \sum_{h=o}^{s-1} \sum_{f=o}^{t-1} \xi_{hf} (\overline{\tau_{hf} \times \underset{-}{\gamma}_h}) (\tau_{hf} \times \underset{-}{\gamma}_h)^T \qquad 5)$$

where ξ_{hf} and τ_{hf} are the fth latent root and vectors of $\underset{-}{\theta}^*_h$. The Moore-Penrose generalized inverse of C is obtained from 5) by substituting β_{hf} for ξ_{hf} where

$$\beta_{hf} = o \quad \text{if} \quad \xi_{hf} = o$$

$$= \xi_{hf}^{-1} \quad \text{otherwise.} \qquad 6)$$

C^- can be written in block circulant form by reversing the above process.

In all cases considered in Chapter 8 C is a symmetric matrix and $\underset{-}{\theta}_k = \underset{-}{\theta}_{s-k}$, $\underset{-}{\theta}^*_h = \underset{-}{\theta}^*_{s-h}$ and the $\underset{-}{\theta}^*_h$ are hermitian matrices. The latent roots of $\underset{-}{C}$ and the $\underset{-}{\theta}^*_h$ are real, ω_h^k may be replaced by $\cos(hk.2\pi/s)$.

CHAPTER 9

THE DSIGNX PROGRAM

9.1 Summary

A computer program DSIGNX is described in which many ideas outlined in Chapters 1 to 7 are implemented. The program can construct a wide range of plans useful in agricultural research and print them in various ways. A brief description of the program is given in §9.2 and the language is described in §9.3.

Effective block and treatment structures help to give DSIGNX flexibility. The interplay between the effective structures and restricted or pseudo-factors is described in §9.4.

The program is controlled by directives which fall into five broad categories: declaration, design construction, randomization, output of experimental plans and miscellaneous. Brief outlines of the directives falling into each category are given in §§9.5 to 9.9. Further information is given in Appendices A9.1 to A9.3.

Appendix A9.4 contains detailed examples of the use of DSIGNX.

9.2 Introduction

The program DSIGNX is intended for statisticians and experimenters engaged in designing comparative experiments and constructing plans to control them. DSIGNX replaces the program DSIGN, written by H.D. Patterson in Orion autocode (c. 1964) and rewritten in FORTRAN by Tolmie (1973), which was in use at Edinburgh and Rothamsted for several years. The new program is designed to be more versatile and easier to use than its predecessor. The programs have some common features e.g. the design key procedure and restricted randomization and both are written in FORTRAN but they differ in

organization and structure and very little, if any, of the original code is used in the new program.

A detailed manual for users has been prepared by Franklin and Mann (1980) and extracts are presented in Appendix A9.1. (A manual for programmers has been prepared but not published.) The range of experimental plans covered by DSIGNX and its usefulness can be fairly assessed from the examples in Appendix A9.4.

In §1.2 steps in the construction of experimental plans are identified. A typical DSIGNX program includes:

1) declaration of block and treatment factors;
2) declaration of the block and treatment structures;
3) declaration of the design;
4) construction of the design;
5) randomization of the units;
6) printing of the experimental plan;

DSIGNX is intended for experiments with simple block structure as defined by Nelder (1965) or for other experiments, such as change-over trials, where the units have similar structural but different randomization properties. The block and treatment structures (§1.3) are defined using formulae similar to those of Nelder (1965) and are used subsequently to control the construction of the design, the randomization and the printing of the experimental plan.

Procedures for constructing the design include cyclic, generalized cyclic (α-series) and design key generators. Each generator may be used with generating arrays held in an associated library or input by the user. The randomization schemes can be modified in various ways including the use of restricted randomization.

There are several forms of output for the experimental plan.

These include a straightforward listing of units and treatments, or some subset, and a two-dimensional field plan which indicates the boundaries of the various blocks. The reported factor levels may be numeric or descriptive. The output can be arranged as an integer array in a card-image file for interfacing with other programs.

The user's instructions are written in free-format style and can be annotated by captions and descriptive headings. The program is intended for interactive use and there is an extensive fault trapping system which enables the user to correct faulty instructions before continuing. It may also be used in batch mode or in a mixture of interactive and batch modes as, for example, when the user transfers control from an interactive terminal to a file containing standard instructions.

9.3 The DSIGNX Language

A program consists of a sequence of statements in the form of directives whose effects may be modified by options. Directives, options and their effects are pre-defined by the system; a list of directives and their functions is given in Appendix A9.2. A directive consists of a directive-name followed possibly by a list of arguments and by option-names which also may be followed by a list of arguments. A directive-name and an option-name must be preceded by a $ and % symbol respectively but the symbol is not part of the name and can be followed by one or more separators (space, newline). Options apply to the preceding directive and arguments to the preceding option or directive. Arguments may be integers, identifiers, character strings, or formulae. Identifiers have up to eight characters, the first being a letter; character strings are a sequence of characters enclosed in quotes; formulae define block and

treatment structures.

Example 9.1:

a) $FACTOR BLOCK 4 PLOT 8

There are no options to directive $FACTOR but the number of arguments can vary. In this case two factors are declared - one, with identifier BLOCK, has four levels and one, PLOT, eight levels.

b) $RANDOMIZE %RRAN PLOT 1

The action of directive $RANDOMIZE is modified by applying a restriction (type 1) to the randomization of levels of factor PLOT.

DSIGNX recognises five types of object which can have identifiers viz. factor, block structure formula, treatment structure formula, design and heading. Identifiers give a distinct name to each object and are defined using declaration directives. An attempt to give identical names to two objects results in either an error message if the objects are of different type or replacement of the first object by the second if of the same type.

Directive-names, option-names, identifiers and formulae must not contain a separator but there must be one or more separators, between each pair of items. Separators are normally ignored by the program to allow versatile input but in character strings they are retained for versatile printing. The symbols $, % and ' are treated as special (and must, for example, be repeated if they are included in character strings). The symbol $ causes input of the preceding directive to be terminated, interpreted and obeyed. It is, in effect, the interaction symbol informing the computer when to commence work. Should any error be detected during interpreting or processing, the effects of the directive are totally ignored and processing restarts at the next $ symbol.

Example 9.2: The following example of a 3×2 factorial in two

randomized complete blocks illustrates the main features of DSIGNX:

```
1)  $FACTOR BLOCK 2 PLOT 6 A 3 B 2
2)  $BSTRUCTURE BLOCK/PLOT        $TSTRUCTURE A*B
3)  $ASSIGN
4)  $RANDOMIZE
5)  $PRHEADING 'EXAMPLE' $
6)  $PRINT $
7)  $END
```

Statement 1) defines the block and treatment factors; 2) specifies the block and treatment structures; 3) allocates treatments to units (implicitly declaring an unnamed design); 4) causes the units to be randomized; 5) prints a heading; 6) causes the randomized plan to be printed in tabular form; 7) terminates the job.

The format of the DSIGNX language is partly modelled on the GLIM language (Baker and Nelder (1978)) as the latter is convenient for statisticians and likely to be known by many users of DSIGNX. The two programs however have different purposes and it is not convenient to adopt the GLIM syntax throughout.

9.4 Plans and designs

A design held within a DSIGNX program can be regarded as a vector with one element for each unit of the plan; the element indicates which treatment is allocated to the unit. Alternatively the design can be regarded as a multi-dimensional table with multi-variate entries, each dimension corresponding to a block factor and each variate to a treatment factor. When a design is declared the form of the table is determined by the block and treatment structure. Space is then allocated for the design and also a vector of length equal to the number of units defined by the block structure. This

vector holds the permutation to be applied to the units in order to derive the experimental plan from the design. (Initially the vector is set for an unrandomized scheme.) For each design only one permutation vector, and hence only one plan, can be held at a time. As the design and the experimental plan are thus directly linked the identifier can be regarded as referring to either. A design can, however, form multiple experimental plans by applying different randomizations.

To minimize references the program defines a current design to which all requests to allocate, randomize, print etc. are applied. The current design is that most recently declared or subsequently set by directive $SETDESIGN. Similarly, one block and treatment structure are defined to be current and any design subsequently declared uses these structures unless others are specified. Any design, block or treatment structure not explicitly referenced need not be given an identifier. Indeed, for most jobs it is necessary to declare only unidentified block and treatment structures for, in the absence of a current design, many directives cause an unnamed design to be implicitly declared using these structures. Any attempt to declare a design when there are no current block and treatment structures causes an error.

In §1.2 units and treatments of the experimental plan are distinguished from unit labels and treatment labels of the design. Similarly the block and treatment structure are distinguished from the effective block and treatment structure. In DSIGNX two operations affect the effective structures

1) restriction ($RESTRICT)

2) pseudo-factors ($PSEUDO, $P)

The effects of the operations is illustrated by Examples 1.7 and 9.3

Example 9.3: Nine treatments, one of which is a control, are allocated to three superblocks each containing two blocks of five units, the control treatment being allocated to one unit in each block. The block and treatment structures for the experimental plan can be written

```
$BSTRUCTURE  SUPERBLOCK/BLOCK/PLOT
$TSTRUCTURE  TREATMENT
```

Treatments other than the control can be allocated in the following way. Restrict units to four in each block by omitting one level of factor PLOT and restrict treatments to exclude the control. The design is then effectively one for eight treatments in three superblocks with two blocks of four units. Applying a quasi-factorial structure to the treatments they can be allocated as for a partially confounded design. The construction of the first superblock involves the following steps:

	restriction	pseudo-factors
Superblock	restrict to one level, omit	none
Block	no restriction	plot factor P_1
Plot	restrict to four levels	plot factors P_2 P_3
Treatment	restrict to eight levels	treatment factors T_1 T_2 T_3

The effective block and treatment structures then contain pseudo-factors P_1, P_2, P_3 and T_1, T_2, T_3 respectively and the appropriate design key matrix has three rows and columns.

Restrictions and pseudo-factors are defined only for the current design. A change in restriction causes a change in pseudo-factors and any change in pseudo-factors causes a change in effective block and treatment structures. Whenever directives $RESTRICT, $PSEUDO or $P are used the effective block and treatment structures are

automatically updated. A switch of current design causes all restrictions and pseudo-factors to be cancelled and the effective block and treatment structures changed to those of the new design.

9.5 Declarations

In this section we describe declaration directives

$FACTOR, $BSTRUCTURE, $TSTRUCTURE, $DESIGN, $HEADING

and allied directives

$SETDESIGN, $DEVALUE, $RESTRICT, $PSEUDO, $P, $LEVELS

The syntax is given in Appendix A9.3.

Each of the five types of declarable object require some basic attribute to be specified at the time of declaration:

Directive	Object	Required attribute
$FACTOR	factor	number of levels
$BSTRUCTURE/$TSTRUCTURE	block/treatment structure	formula
$DESIGN	design	block and treatment structures
$HEADING	heading	character string.

The number of levels for a factor is a positive integer. The formal levels held within the program are 0, 1 ... n-1, where n is the number of levels, but the input and output levels are 1, 2 ... n. The block and treatment formulae are a subset of those described by Wilkinson and Rogers (1973). Operators allowed in the block structures are nesting (/) and crossing (*); in treatment structures the operator union (+) is allowed also. All factors in the formulae must be declared. The size of a design is determined by the associated block and treatment structures which must have been declared previously. The number of units is the product of the block factor levels. The numbers of treatment factor levels are required

by the program but operators in the treatment structure formula are not explicitly used. Identified headings can be printed by directive $PRHEADING.

Normally the current design is that most recently declared but a new one may be specified by directive $SETDESIGN. Designs can be deleted and space recovered by use of directive $DEVALUE; headings also can be deleted.

Directives $RESTRICT, $PSEUDO, $P, and $LEVELS are concerned with the formation of the effective block and treatment structures and are interlinked (§9.4). Directive $RESTRICT restricts the levels of selected factors to lie between specified values a and b ($a \leq i \leq b$) but may also cancel the restrictions. Directive $PSEUDO is a flexible pseudo-factor routine and can cancel current pseudo-factors etc; directive $P is a simple routine which replaces all block and treatment factors by pseudo-factors, one for each prime component of the number of levels of the original factor. Both directives $PSEUDO and $P apply to the design as currently restricted; a subsequent call to directive $RESTRICT causes pseudo-factors associated with factors named to be cancelled. Directive $LEVELS lists the number of levels of factors or pseudo-factors in the effective block and treatment structures.

Directive $RESTRICT has two distinct roles. If applied before construction, the design can be formed from combining sub-designs; if applied after construction the design can be formed by deleting factor levels in an existing design. It may also be used to restrict randomization of units and treatments.

9.6 Constructing the design

The directives described in this section fall into two categories, those which directly allocate treatments to units (i.e. treatment labels to unit labels) and those which combine two designs to form a third. The four directives in the first category are

$ASSIGN, $DESKEY, $CYCLIC, $ALPHA

and the three in the second category are

$MERGE, $PRODUCT, $JOIN.

The syntax is given in Appendix A9.3.

All these directives are affected by the current restrictions but only $DESKEY by pseudo-factors. When a factor is restricted to levels a, a+1 ... b inclusive the program constructs the design as if the factor has levels 1 to b-a+1 inclusive, and subsequently converts levels to lie within the range of restriction. The operations are therefore dependent only on the number of levels of the restricted factors and not on the particular levels chosen.

For all directives the design is constructed by processing units in lexicographical order as defined by the effective block structure and determining for each unit the treatment to be allocated. The seven directives correspond to different allocation rules.

9.6.1 Directives $ASSIGN and $ALPHA

Directive $ASSIGN optionally allows treatments to be input through the user's program in the form of list and is commonly useful when designs are required which cannot be generated by any other directive. The default option causes the treatments to be generated in lexicographical order, the sequence being repeated as often as required to exhaust the units.

Directive $ALPHA constructs generalized lattice (α-series)

designs. The generating arrays for the designs may be obtained optionally (%PWH and %ER) from libraries of tables originally produced by Patterson, Williams and Hunter (1978) or Williams (1975) (see also Appendix A9.1); the arrays required are determined automatically from the block structure. A third option (%READ) allows the generating array to be input through the user's program.

9.6.2 Directive $CYCLIC

This directive generates cyclic and allied designs (§2.4). The options (%JWD, %DH and %HW) generate designs from stored initial blocks originally given by John, Wolock and David (1972), Davis and Hall (1969) or Hall and Williams (1973) respectively (see Appendix A9.1). Another option (%READ) allows the user to input initial blocks of his choice and can be used (with option %INCREMENT) to yield generalized cyclic designs as described by Jarrett and Hall (1978). Options %HW and %READ allow the design to have more than one treatment factor but at present the factors must have the same number of levels. Levels in initial blocks not corresponding to any treatment cause an error but levels corresponding to treatments excluded by current restrictions are treated as invariant and not affected by the cyclic process.

Example 9.4: A BIB (6, 5, 3, 10; 2) can be constructed by restricting the treatment factor to levels 1 to 5 inclusive and generating the design from initial blocks (1 2 4) and (1 2 6):

1	2	3	4	5		1	2	3	4	5
2	3	4	5	1		2	3	4	5	1
4	5	1	2	3		6	6	6	6	6

9.6.3 Directive $DESKEY

This directive generates a factorial design through a design key

generator controlled by the effective block and treatment structures. There are six main options, five of which are used to obtain a standard key matrix from a disk file and the sixth allowing the user to input the required matrix:

%FRACTION	fractional two- and three-level factorial designs
%FACTORIAL	confounded designs
%LATINSQ	orthogonal Latin squares
%LATTICE	lattice designs
%LASQUARE	lattice square designs
%READ	input design key matrix.

The standard fractional designs are similar to those discussed in §3.4 and cover most useful fractional/confounded two- and three-level factorial designs. The confounded designs are isomorphic under randomization to those of Cochran and Cox (1957, Ch.6). The Latin square designs and lattice designs are closely related, there being one superblock in the lattice design for each Latin square plus two other superblocks. The order of the superblocks is as described in §2.6.1 and as shown there ϕ-series designs are readily constructed from the lattice designs by applying suitable restrictions (when randomizing and printing).

Correct use of a standard design key requires that the effective block and treatment structures are compatible with the key matrix in the number of (pseudo) factors, the numbers of levels of the factors and their ordering. Standard matrices normally construct a single superblock using a key matrix applicable to prime-level (pseudo) factors with the superblock factor suppressed. (The correct ordering is indicated in the manual and can be checked by directive $LEVELS.)

Option %BASE causes a specified 'base' treatment vector to be added for each unit allowing, for example, a fractional design to

be formed from other than the principal block.

9.6.4 Directives $MERGE, $PRODUCT and $JOIN

Directive $MERGE forms a design by merging or 'adding' two other designs (§7.2.4). It is useful for constructing, say, a factorial design with treatment structure A*B*C + A*D from designs with treatment structures A*B*C and A*D . The block structure of the three designs must differ in at most one category and for this category the number of levels in the current design is the sum of the levels in the other two designs.

Example 9.5: Let DA, DB and DC be three designs with block structures

SUPERBLOCK/BLOCK/PLOT_A

SUPERBLOCK/BLOCK/PLOT_B

SUPERBLOCK/BLOCK/PLOT_C

respectively, then designs DA and DB may be merged to form design DC only if the (restricted) number of levels of factor PLOT_C is the sum of the levels for factors PLOT_A and PLOT_B.

The treatment structure of the current design must contain all factors occurring in the treatment structures of the other two designs.

Note: Any design constructed using directive $MERGE can also be constructed using directive $RESTRICT without recourse to multiple designs.

Directive $PRODUCT forms the current design from the 'product' of two other designs (§7.2.4). It is useful for superimposing a design upon the units of an existing design e.g. converting a randomized block design to a split plot design. If the block structures of the initial designs are BSA and BSB respectively then the current design must have (restricted) block structure

BSA/BSB or BSA*BSB .

Treatments in the current design are combinations of those in the original designs.

Directive $JOIN forms a design from two others with identical block structure by combining the treatments. It is useful for superimposing extra treatment factors on an existing design (§7.2.4).

Option %KEEPRANDOM: Directives $MERGE, $PRODUCT and $JOIN contain this option which causes the permutation vector of the current design to be constructed from those of the initial designs, but the effect is different for each.

9.7 Randomization directives

Three directives relate to randomization procedures:

$RANNUMBER, $RANDOMIZE and $RANTREATMENTS.

The first directive enables the pseudo-random number generator to be reset or printed for future reference. The last two directives cause randomization of units and treatments respectively.

The randomization of units is normally controlled entirely by the block structure formula as outlined in §7.3 (and may be applied before treatments are allocated). The user may modify this process through option %RRANDOM by restricting, or totally suppressing, randomization of levels for some factors. Restricted randomization is allowed for factors with eight, nine or 16 levels using the procedures of Grundy and Healy (1950) and Dyke (1964) as modified in §7.4. Randomization of treatments causes the levels of selected treatment factors to be permuted at random.

The effects of both directives $RANDOMIZE and $RANTREATMENTS can be modified by restrictions imposed by directive $RESTRICT. Such randomization allows control treatments to be unaffected. (See e.g. Patterson, Williams and Hunter (1978, p. 397.))

9.8 Output of experimental plan

Eight directives, together with directives $C, $OUTPUT, and $ECHO (§9.9), permit a variety of output forms for the experimental plan and the user's program. The directives are:
$LAYOUT, $NAMES, $SUPPRESS, $NUMBER, $PRINT, $MAP, $PRHEADING, $PAGE.

Two distinct forms of output for the experimental plan are:

1) that produced by directive $PRINT in which the plan is a listing of all units and treatments allocated to them;

2) that produced by directive $MAP in which the experimental plan is produced as a two-dimensional (field) plan.

The syntax is given in Appendix A9.3.

9.8.1 Directives $LAYOUT, $NAMES, $SUPPRESS and $NUMBER

The two-dimensional plan requires that each block is constructed as a rectangular array of experimental units, the dimensions of which has been given in a previous call of directive $LAYOUT. This directive, besides storing the block dimensions, also checks that they are consistent with the block structure.

In both forms of output the experimental plan can be printed with numeric or descriptive factor level names. The names are input as character strings using directive $NAMES and used if option %PRNAMES is requested within directives $PRINT or $MAP. Directive $SUPPRESS is used in conjunction with directive $NAMES to partially suppress printing for some treatments.

Normally units are numbered from 1 in lexicographical order; when some other sequence is required directive $NUMBER is used. Options allow numbers to run right across rows or columns of the design if directive $LAYOUT has been used. Another option (%READ) allows the user to input unit numbers.

9.8.2 Directives $PRINT and $MAP

Directive $PRINT prints the experimental plan as a list in lexicographical order of units. In the default printing there is a column containing unit numbers and a column for each block and treatment factor. Descriptive captions are also given.

Apart from the use of descriptive names printing can be varied in several other minor ways. One option (%FORMS) allows the list to be superimposed on primitive recording forms. Options (%NOCAPTIONS and %NONUMBERS), shared with directive $MAP, suppress captions and unit numbers. (The former option is useful when interfacing with another program because the output plan consists of an array of integers.) Other options (%NOBLOCKS and %NOTREAT) suppress printing of block factors or treatment factors.

In the two-dimensional field plan, boundaries of blocks corresponding to the ith factor in the block structure are indicated by digit i but where two boundaries coincide only the smaller value is printed; the complete plan is bordered by zeros. These borders can be changed optionally to show the nesting depth of the associated factor in the block structure (%DEPTH) or suppressed to give a more compact output (%COMPRESS). (Suppression of both boundaries and captions makes this form of output also useful for an interface.)

Output from directive $MAP is commonly too large to fit onto one one page of computer paper. The plan can be subdivided by rows (%ROWS) or by columns (%COLUMNS) or both and printed in sections; it can also be rotated through 90^{o} (%ROTATE).

Directive $RESTRICT can be used with both directives $PRINT and $MAP to restrict printing to part of the plan.

9.8.3 Directives $PRHEADING and $PAGE

Directive $PRHEADING prints an identified heading or a character string. If used with neither it causes a line to be skipped. $PAGE causes printing to commence on a new page.

9.9 Other directives

The directives described in this section can be divided roughly into two categories, those related to management of input and output channels and those related to management of the program itself. Those in the former category are:

$INPUT, $OUTPUT, $LABEL, $REWIND, $ECHO,

and those in the latter category are:

$C, $ENVIRONMENT, $DEBUG, $NEXT, $END, $STOP.

9.9.1 Channel control

In many applications of DSIGNX secondary channels are required for input and output. Secondary input channels are useful when reading standard sets of instructions or standard designs; secondary output channels are useful when storing plans or linking to a line printer following work on an interactive terminal. DSIGNX allows for several input and output channels.

During execution of a program instructions are read from the current input channel and output is directed to the current output channel. At the start of a session these are normally the card reader and line printer when in batch mode, or the terminal when in interactive mode. For convenience, the initial channels are referred to as the primary input channel and the primary output channel. (Error messages, warnings and printing of the user's program, if requested, are directed to the primary output channel.)

New current input and output channels are selected through

directives $INPUT and $OUTPUT which may also specify the line width. Neither channel is rewound during the switch and if the original is used again processing normally starts at the next record. Directives $REWIND and $LABEL can be used to determine other starting positions. (These directives allow the construction of primitive macros which are, in effect, identified by the label.)

Directive $ECHO controls the printing of input records. Normally printing is required when operating in batch mode but not interactive mode.

9.9.2 Miscellaneous directives

Directives $ENVIRONMENT and $DEBUG are intended to meet somewhat different requirements but can be used together to detect program errors. Directive $ENVIRONMENT lists details of the particular DSIGNX installation including special symbols, permitted channel numbers and their widths, maximum number of factors, etc. Also given is information about the state of the program such as the number of factors used to date, amount of main store used, etc. This information is commonly sufficient to expose minor errors. Directive $DEBUG provides more detailed information but assumes a greater degree of knowledge about the program and gives, for example, details of internal pointers and links.

Directive $NEXT causes all stores and pointers within the program to be cleared in preparation for starting a new program. Directives $END and $STOP are synonyms and cause the program to be terminated.

Appendix A9.1

Extracts from the DSIGNX Users' Manual

This page and the following three pages contain extracts from the DSIGNX Users' Manual. They illustrate the style of the manual and give extra information on some topics discussed in this thesis.

2. THE DSIGNX PROGRAM

2.1 Purpose

The DSIGNX package was written to facilitate the construction of plans for use in designed experiments. Stages in the construction of experimental plans are distinguished and various methods of performing each stage are given. Experimental plans are put to a variety of uses and so a range of procedures has been provided. The DSIGNX program is likely to prove most useful in large experiments or for experiments involving several trials based on similar designs.

2.2 Mode of use

The DSIGNX program may be used in either an *interactive* or a *batch* mode, depending upon the particular installation. In both modes instructions are acted upon as they are read and the results printed out or, if an error occurs, a message is printed. In interactive mode error messages are always output on the user's terminal. The user may then correct and re-input the offending instruction or take any other action he feels necessary. In batch mode the message is printed but processing of the instructions continues.

Sometimes an error will cause others to occur e.g. an incorrectly specified factor will cause errors whenever an attempt is made to use this factor. However, continued processing often helps to uncover other errors and so helps to reduce programming time.

When the program is being used interactively, it is often convenient to gain access to a standard set of instructions. These instructions may be stored in a file which is placed on a different input channel to the user's terminal. When the user transfers control to the alternative channel the program works in batch mode until control is returned to the terminal.

2.3 The character set

The following characters are used in DSIGNX instructions:

Letters	A B C ... Y Z
Digits	0 1 2 3 4 5 6 7 8 9
Separators	*space newline*
Operators	+ * /
Brackets	( )

Appendix B3. Standard Cyclic Designs

The standard cyclic designs held within DSIGNX come from three sources, the tables of John et al (1972), the cyclic change-over designs presented by Davis and Hall (1969) and the cyclic superimposed designs presented by Hall and Williams (1973). The designs of John et al are too numerous to detail here. The initial blocks for the other designs are presented below. The user should refer to the original publications for the properties of the designs.

i) Cyclic change-over designs

	k = 3 periods		k = 4 periods		k = 5 periods	
t	reference	initial blocks	reference	initial blocks	reference	initial blocks
6	1	(034) (051)	16	(0132) (0314)	31	(01325)
7	2	(031) (045)	17	(0136) (0641)	32	(02315)
8	3	(041) (065)	18	(0214) (0153)	33	(01325)
9	4	(038) (067)	19	(0142) (0526)	34	(01325)
10	5	(013) (054)	20	(0421) (0574)	35	(03187)
11	6	(017) (0102)	21	(0512) (0652)	36	(04712)
12	7	(017) (0113)	22	(01114) (01034)	37	(01547)
13	8	(014) (0116)	23	(0139) (0869)	38	(02374)
14	9	(019) (0133)	24	(0715) (0643)	39	(034115)
15	10	(082) (0144)	25	(0715) (051112)	40	(01547)
16	11	(069) (0415)	26	(0571) (014613)	41	(08561)
17	12	(015) (0159)	27	(0175) (0329)	42	(02934)
18	13	(015) (0122)	28	(0185) (011212)	43	(091287)
19	14	(0418) (0177)	29	(0769) (0121611)	44	(07389)
20	15	(0213) (0194)	30	(01186) (01947)	45	(01429)

ii) Cyclic superimposed designs

	k = 4		k = 5		k = 6	
t	reference	initial blocks	ref.	initial blocks	ref.	initial blocks
5	1	(0123,1302)				
6	2	(0124,0241)	12	(01234,02413)	22	(012345,532140)
7	3	(0124,0241)	13	(01234,04321)	23	(012345,135024)
8	4	(0125,0251)	14	(01235,02153)	24	(012346,036214)
9	5	(0134,1430)	15	(01236,02163)	27	(012346,036214)
10	6	(0125,1520)	16	(01236,10632)	26	(012357,027531)
11	7	(0125,1520)	17	(01247,90261)	27	(012457,147250)
12	8	(0137,1730)	18	(01247,071210)	28	(012368,038216)
13	9	(0139,15011)	19	(01269,61920)	29	(012359,021593)
14	10	(0146,0461)	20	(01269,61920)	30	(0123610,1362010)
15	11	(01310,11030)	21	(01249,09421)	31	(012379,039217)

B4. Generalized lattice (alpha series) designs

The generating arrays for generalized lattice (alpha series) designs held within DSIGNX come from two sources, arrays for $s \leq k$ presented by Patterson, Williams and Hunter (1978) and arrays for two-replicate designs $s \geq k$ presented by Williams (1976). The arrays are listed below and are suitable for $v \leq 100$ variables. In each case the first row comprising entirely of zeros has been omitted.

$s \geq k$

These arrays are suitable for two, three or four replicate designs which have block size k less than or equal to the smaller of s and the integer part of 100/s and s ranging from 5 to 15, ($k \geq 4$).

s = k = 5

```
0 1 2 3 4
0 4 3 2 1
0 2 4 1 3
```

s = k = 6

```
0 1 3 2 4 5
0 5 2 3 1 1
0 4 5 1 2 3
```

s = k = 7

```
0 1 2 4 3 5 6
0 3 6 5 2 1 4
0 2 4 1 6 3 5
```

s = k = 8

```
0 1 3 5 2 4 6 7
0 2 7 3 5 1 0 6
0 6 1 4 3 6 2 5
```

s = k = 9

```
0 1 3 7 2 4 5 6 8
0 8 6 2 3 1 7 5 4
0 7 4 3 5 6 2 1 7
```

s = k = 10

```
0 1 3 5 4 6 7 8 9 2
0 9 6 7 5 3 2 4 8 6
0 5 9 2 6 1 4 7 2 3
```

s = 11, k = 9

```
0 1 4 9 2 5  6 3 7
0 6 8 7 3 1  5 9 4
0 7 1 5 6 3 10 4 1
```

s = 12, k = 8

```
0 1 7 9  4 11 10 5
0 2 5 6 11  3  4 1
0 3 1 4  8 10  7 6
```

s = 13, k = 7

```
0  1  3 9 12  8 6
0  4  8 2 10  5 7
0 10 11 1  6 12 8
```

s = 14, k = 7

```
0  1  9 11 2  5  3
0  8 10 13 6 11  1
0 10  7  2 1 12 11
```

s = 15, k = 6

```
0 1  3 7 10 14
0 8 12 2 13  3
0 9 14 5 11  8
```

APPENDIX C: ERROR CODES AND MESSAGES

The error codes are divided into two sets, those which may occur in several directives (e.g. syntax) and those which are specific to a given routine. The message printed when an error occurs takes the form

```
* FAULT nn IN aaaa - message
```

where aaaa is the name of the last directive read.

Those errors which are specific to a given directive are numbered 21 upwards while those which are not specific are numbered from 1-20.

Only one error is recognised in any one directive; if an error occurs the rest of the directive is ignored. The program will skip to the next directive and continue, so if a directive (e.g. a declaration) fails it could cause other directives to fail as well.

Fault No.	Message	Action by interactive user
1	Invalid character: syntax error	The next item to be input does not agree with the syntax. Check the input for the current directive.
2	Invalid integer	An integer has not been found when expected or it is outside the range permitted.
3	Number list wrong length	Check the length of the integer list.
4	Invalid directive	Check the spelling of the directive name.
5	Invalid option	Check the spelling of the options requested and whether the option is available for the current directive.
6	Identifier already used	An identifier has been used previously for a different type. Use a different identifier.
7	Too many identifiers of this type	Check the number of identifiers of the current type. Re-use some other identifiers if possible.
8	Too many identifiers	The space for holding identifiers and the related information has been exhausted. Re-use identifiers if possible.
9	Invalid factor	An identifier is not that of a factor. Check the spelling.
10	Factor not in block structure	Check that the specified block factors are in the block structure of the current design.

Rest of Appendix C omitted

Appendix A9.2

The DSIGNX directives

The 42 directives are placed in broad categories according to their function. They are described in more detail in Appendix A9.3.

Category: 1 = declaration and allied directives, 2 = construction of design, 3 = randomization, 4 = output of experimental plan, 5 = channel control, 6 = other directives.

Directive	Category	Remark
$ALPHA	2	α-series design generator
$ASSIGN	2	Input or standard design generator
$BFACTOR	1	Block factor declaration
$BSTRUCTURE	1	Block structure declaration
$C	6	Comments
$CYCLIC	2	Cyclic design generator
$DEBUG	6	Program debugging
$DESIGN	1	Design declaration
$DESKEY	2	Design key generator
$DEVALUE	1	Design and heading deletion
$ECHO	5	Printing instructions
$END	6	Program termination
$ENVIRONMENT	6	Installation information
$FACTOR	1	Factor declaration
$HEADING	1	Heading declaration
$INPUT	5	Input channel selection
$JOIN	2	Treatment superimposition
$LABEL	5	Input channels labelling
$LAYOUT	4	Field plan definition
$LEVELS	1	Effective block and treatment structures
$MAP	4	Field plan printing
$MERGE	2	Adding designs
$NAMES	4	Factor level naming
$NEXT	6	Program initialization
$NUMBER	4	Unit numbering
$OUTPUT	5	Output channel selection
$P	1	Forming prime-level pseudo-factors
$PAGE	4	Page changing
$PRHEADING	4	Heading printing
$PRINT	4	Experimental plan listing
$PRODUCT	2	Nesting or crossing designs
$PSEUDO	1	Forming pseudo-factors
$RANDOMIZE	3	Units randomization
$RANNUMBER	3	Setting/printing random number
$RANTREATMENT	3	Treatments randomization
$RESTRICT	1	Design restriction
$REWIND	5	Channel control
$SETDESIGN	1	Design selection
$STOP	6	Program termination
$SUPPRESS	4	Printing suppression
$TFACTOR	1	Treatment factor declaration
$TSTRUCTURE	4	Treatment structure declaration.

Appendix A9.3

Syntax of DSIGNX directives

In this appendix the syntax of the DSIGNX program is summarised. Square brackets [ ] indicate an entry is optional and may be omitted, angular brackets < > indicate that there may be several entries of the same type. All options may be omitted.

Declaration and related directives

$FACTOR (alternative $BFACTOR, $TFACTOR)

declares identifiers and numbers of levels for one or more factors.

Syntax: $FACTOR <identifier integer>

where identifier is that of a factor and integer is the number of levels.

$BSTRUCTURE

declares one or more block structures.

Syntax: $BSTRUCTURE <[identifier] formula>

where identifier is that of a block structure and formula is based on that described by Nelder (1965) and may contain operators * and / .

$TSTRUCTURE

declares one or more treatment structures.

Syntax: $TSTRUCTURE <[identifier] formula>

where identifier is that of a treatment structure and formula is based on a subset of those of Wilkinson and Rogers (1973) and may contain operators *, / and + .

$DESIGN

declares a design by defining the block and treatment structures.

Syntax: $DESIGN [identifier D] %BLOCK identifier B %TREATMENT identifier T where identifier D is that of the design.

Options %BLOCK identifier B

%TREATMENT identifier T

where identifier(s) define block/treatment structures. If either structure is current the option may be omitted.

Note: A call on any of the directives

$ASSIGN, $CYCLIC, $ALPHA, $DESKEY, $PSEUDO, $P, $LEVELS
$RESTRICT, $RANDOMIZE, $LAYOUT, $NUMBER, $SUPPRESS

causes, if there is no current design, an implicit call on directive $DESIGN with arguments and options omitted.

$HEADING

declares one or more headings and their identifiers.

Syntax: $HEADING <identifier 'character string'>

where identifier is that of a heading and character string contains the heading contents.

$SETDESIGN

designates a new current design.

Syntax: $SETDESIGN identifier

where identifier is that of a design.

$DEVALUE

deletes designs and headings and allows space recovered to be reused.

Syntax: $DEVALUE <identifier>

where identifier is that of a design or heading.

Note: unidentified designs which are not current are deleted.

$PSEUDO

converts some factors in the current design to pseudo-factors.

Syntax: PSEUDO <identifier [<integer>]>

where identifier is that of a factor and integer is the number of levels of a pseudo-factor. If <integer> is omitted prime-levels are assumed.

$P

converts all factors in the current design to prime-level pseudo-factors.

Syntax: $P

$RESTRICT

causes the levels of selected block and treatment factors in the current design to lie between specified levels.

Syntax: $RESTRICT <identifier integer a integer a'>

where identifier is that of a factor, integer a is the lowest level and integer a' the highest ($1 \leq a \leq a' \leq$ no. of levels of the factor).

Note: If integer a' = 0 the factor is restricted to level (integer a) and omitted from the effective block or treatment structure.

$LEVELS

prints levels of the (pseudo) factors in the effective block and treatment structures.

Syntax: $LEVELS

Design construction directives

$ALPHA

generates generalized lattice (α-series) designs.

Syntax: $ALPHA %READ <integer> %PWH %ERW

The three options are alternatives.

Options %READ <integer> where integer is an element of an r×k generating array input by rows; the design is constructed from the array.

%PWH
%ERW cause the design to be constructed from stored arrays based on tables presented by Patterson, Williams and Hunter (1978) and Williams (1975). (See also Appendix A9.1).

The size of the array is determined by the block structure.

$ASSIGN

constructs the experimental design either by a simple generator or by a list input through the user's program.

Syntax: $ASSIGN %READ <integer>

The default option generates treatments in lexicographical order.

Option %READ <integer> where integer is the level of a treatment factor; the design is formed from the list which contains factor levels for one unit followed by those for the next unit etc. Elements in the list corresponding to levels of the ith factor satisfy $1 \leq$ integer $\leq a'_i - a_i + 1$ where a_i, a'_i are the lowest and highest levels of the (restricted) ith treatment factor.

$CYCLIC

generates cyclic and allied designs.

Syntax: $CYCLIC %JWD integer A integer B %DH integer %HW integer %READ <integer> %KVAL integer %INCREMENT integer

Options %JWD, %DH, %HW and %READ are alternatives; option %INCREMENT applies only when option %READ is used.

Options %JWD integer A integer B where integer A and integer B reference a design in tables of John, Wolock and David (1972). Tables A, B, C and D are indicated by values 1, 2, 3 and 4 for integer A. The design number is integer B. The design is constructed from initial blocks in the tables.

%DH integer where integer is the reference number of a cyclic change-over design presented by Davis and Hall (1969). The design is constructed from initial blocks given in the tables.

%HW integer where integer is the reference number of a cyclic superimposed design presented by Hall and Williams (1973). There must be two treatment factors with equal number of levels. The design is constructed from initial blocks given in the tables.

%READ <integer> where integer is an element of an initial block. All treatment factors have equal number of levels, t. For u factors, i initial blocks and block size k initial blocks are read as u arrays of dimension $i \times k$ read successively by rows. The design is formed from the initial blocks.

%INCREMENT integer causes integer, which must be a divisor of t, to be added to each treatment level in successive steps of the cycle. The default value is 1.

%KVAL integer sets the length of initial blocks to integer. The default value is the number of levels of the last factor in the block structure formula.

$DESKEY

generates a factorial design by the design key procedure described by Patterson (1976).

Syntax: $DESKEY %(standard) integer A integer B %READ <integer>
%BASE <integer>

Options %(Standard) and %READ are alternatives. Options %(Standard) are

%FRACTION %LATIN %FACTORIAL %LATTICE %LASQUARE.

Options %FRACTION integer A integer B where integer A references a standard array (§3.5) and integer B is redundant. The option selects a standard design key matrix for a 2- or 3-level fractional factorial design with confounding. The matrix dimensions are determined by the effective block and treatment structures.

%LATIN	integer A	integer B
%FACTORIAL	integer A	integer B
%LATTICE	integer A	integer B

%LASQUARE integer A integer B where integer A defines a standard set of design key matrices and integer B a member of the set. The options select a key matrix suitable for generating one (orthogonal) Latin square or one super-block of a confounded, lattice or lattice square design.

%READ <integer> where integer is an element of a design key matrix read by rows. The size of the matrix is determined by the effective block and treatment structures.

%BASE <integer> inputs a vector whose ith element is added to the level of the ith treatment (pseudo) factor.

$JOIN

forms the current design by superimposing the treatments of two designs with the same block structure.

Syntax: $JOIN identifier A identifier B %KEEPRANDOM
where identifier A and identifier B are those of designs to be combined.

Option %KEEPRANDOM sets the randomization of the current design to that of the design with identifier A.

$MERGE

forms the current design by merging units of two other designs.

Syntax: $MERGE identifier A identifier B %KEEPRANDOM
where identifier A and identifier B are those of designs to be merged.

Option %KEEPRANDOM sets the randomization of the current design to that obtained by randomizing the two sub-designs separately.

$PRODUCT

forms a design by nesting or crossing two other designs.

Syntax: $PRODUCT identifier A identifier B %KEEPRANDOM
where identifier A and identifier B are those of designs to be combined.

Option %KEEPRANDOM sets the randomization of the current design to that for BSA*BSB where BSA and BSB are original block structures.

Randomization

$RANNUMBER

resets or prints the value of the pseudo-random number generator.

Syntax: $RANNUMBER [integer A [integer B [integer C]]]
where $0 \leq$ integer A, integer C $< 2^{12}$, $0 \leq$ integer B $< 2^{11}$. The pseudo-random number is in three parts and can be changed totally or in part; the final value is always printed.

$RANDOMIZE

causes units to be randomized as determined by the block structure.

Syntax: %RANDOMIZE %RRAN <identifier integer>

Option %RRAN identifier integer where identifier is that of a block factor and integer indicates the form of restriction. The option restricts randomization of levels of specified factors in four ways: integer = 0) no randomization; 1) eight-level factors; 2) 16-level factors; 3) nine-level factors. Randomization 1) and 3) are described by Grundy and Healy (1950) and 2) by Dyke (1964). This option causes restrictions to be placed on the randomization of the levels of named factors.

$RANTREATMENT

randomizes levels of treatment factors.

Syntax: $RANTREATMENTS <identifier>

where identifier is that of a treatment factor in the current design.

Output of experimental plan

$LAYOUT

specifies dimensions of the current experimental plan and its blocks in preparation for printing it as a two-dimensional field plan.

Syntax: $LAYOUT integer R integer C <identifier integer R integer C>

where identifier is that of a block factor; integer R and integer C are numbers of rows and columns of experimental units in the overall plan or in blocks defined by the associated block factor.

$MAP

prints the current experimental plan as a two-dimensional field plan as determined by directive $LAYOUT.

Syntax: $MAP %PRNAMES %NONUMBERS %NOCAPTIONS %DEPTH %COMPRESS %COLUMNS integer %ROWS integer %ROTATE

Options %PRNAMES substitutes descriptive factor level names defined using directive $NAMES.

%NONUMBER

%NOCAPTIONS suppress unit numbers and descriptive captions.

%DEPTH prints block boundaries with digits indicating nesting depth of the associated factors. The option has no effect in completely nested designs.

%COMPRESS compresses output by omitting block boundaries.

%COLUMNS integer

%ROWS integer divide the field plan into sections for printing. The location of divisions is determined by integer.

%ROTATE causes the experimental plan to be rotated through 90°.

$NAMES

assigns descriptive names to factors and their levels.

Syntax: $NAMES <identifier ['string A'][integer <'string B'>]>

where identifier is that of a factor; string A is the name (to be printed) of the factor; integer is the number of characters in the level names; string B is a level name.

$NUMBER

assigns numbers to units.

Syntax: $NUMBER %READ <integer> %BYROWS %BYCOLUMNS

The options are alternatives and the default option numbers units in lexicographical order.

Options %READ <integer> where integer is a unit number. The list contains one number for each unit defined by current restrictions.

%BYROWS
%BYCOLUMNS number the units by rows or columns of a field plan defined in a previous call of directive $LAYOUT.

$PAGE

causes output to continue to a new page.

Syntax: $PAGE

$PRHEADING

prints one or more headings in succession.

Syntax: $PRHEADING [<heading>]

where heading is a character string or a heading identifier. If the heading list is omitted then a blank line is printed.

$PRINT

prints the experimental plan as a list of units and treatments.

Syntax: $PRINT %PRNAMES %NONUMBERS %NOCAPTIONS %NOBLOCKS %NOTREATMENTS %FORMS [<integer>]

Options %PRNAMES substitutes descriptive factor level names defined using directive $NAMES.

%NONUMBER
%NOCAPTIONS suppress unit numbers and descriptive captions.

%NOBLOCKS
%NOTREATMENTS suppress block and treatment factor levels.

%FORMS[<integer>] superimposes the list onto a simple recording form where the ith integer determines the width of the column required for the ith entry.

If the integer list is omitted a standard form is printed.

$SUPPRESS

determines the suppression of some treatment names when option %PRNAMES is used in directives $PRINT and $MAP.

Syntax: $SUPPRESS identifier A <integer <identifier B>>

where identifier A is that of a factor controlling printing; integer is a level of this factor; identifier B is a factor which is suppressed whenever factor A has level integer.

Channel control

$ECHO

reverses current instructions on printing input records.

Syntax: $ECHO

$INPUT
$OUTPUT select new input/output channels.

Syntax: $INPUT integer A [integer B][label]
$OUTPUT integer A [integer B]

where integer A is the number of the new channel; integer B is the line width; label is a label indicating where processing is to start. If label is omitted processing starts at the next available record.

$LABEL

sets a label within instructions held on a secondary input stream.

Syntax: $LABEL label

where label has the same form as an identifier.

$REWIND

rewinds channels other than current or primary input/output channels.

Syntax: $REWIND integer

where integer is a suitable channel number.

Other directives

$C

causes all subsequent text up to the next $ symbol to be ignored.

Syntax: $C .

$DEBUG

prints information on the contents of directories, work areas etc.

Syntax: $DEBUG

$ENVIRONMENT

prints information about the environment in which the program is working and the numbers of factors etc. so far declared.

Syntax: $ENVIRONMENT .

$NEXT

clears all work areas, directories, pointers etc. in preparation for a new set of instructions.

Syntax: $NEXT

$STOP
$END terminate the session.

Syntax: $END, $STOP.

Syntax: $LABEL label

where label has the same form as an identifier.

$REWIND

rewinds channels other than current or primary input/output channels.

Syntax: $REWIND integer

where integer is a suitable channel number.

Other directives

$C

causes all subsequent text up to the next $ symbol to be ignored.

Syntax: $C .

$DEBUG

prints information on the contents of directories, work areas etc.

Syntax: $DEBUG

$ENVIRONMENT

prints information about the environment in which the program is working and the numbers of factors etc. so far declared.

Syntax: $ENVIRONMENT .

$NEXT

clears all work areas, directories, pointers etc. in preparation for a new set of instructions.

Syntax: $NEXT

$STOP
$END terminate the session.

Syntax: $END, $STOP.

Appendix A9.4

Examples of the use of DSIGNX

Some examples of the use of DSIGNX are presented. They are chosen to highlight various features of the program and are not necessarily the only or best way of producing the experimental plan.

Examples

1	-	A 2*3 factorial in two randomized blocks
2	-	Split-plot design with different forms of output
3	-	Automatic design key (fractional design)
4	-	Printing under restriction - a partially confounded (2**3)*3 in blocks of six plots from a (2**5) designs in blocks of eight plots
5	-	Design with mixed treatment set formed by restricting; also selective suppression of treatment level names
6	-	Design with mixed treatment set formed by merging two smaller designs
7	-	Adding subplots to an α-series design using $PRODUCT
8,9	-	Adding an extra treatment factor using $JOIN
10	-	Restricted randomization.

```
   $C EXAMPLE 1 - A 2*3 FACTORIAL IN TWO RANDOMIZED BLOCKS

      DEFINE THE ARRANGEMENTS FOR BLOCKS AND TREATMENTS
$FACTOR BLOCKS 2 PLOTS 6  N 3 P 2
$BSTRUCTURE BLOCKS/PLOTS
$TSTRUCTURE N*P
   $C AT THIS POINT THE DESIGN CAN BE EXPLICITLY DECLARED USING THE
      'DESIGN' DIRECTIVE BUT IT IS NOT NECESSARY IF THERE IS ONLY ONE
      DESIGN IN THE JOB
$DESIGN
   $C ASSIGN THE TREATMENTS TO THE PLOTS IN 'STANDARD' ORDER
$ASSIGN
   $C RANDOMIZE THE DESIGN
$RANDOMIZE
   $C PRINT THE DESIGN, STARTING WITH A HEADING
$PRHEADING 'EXAMPLE 1
2-REPLICATE 3*2 DESIGN' $
EXAMPLE 1
2-REPLICATE 3*2 DESIGN
$PRINT   $

UNIT  B1  B2      T1  T2
   1   1   1       1   1
   2   1   2       3   2
   3   1   3       2   2
   4   1   4       1   2
   5   1   5       3   1
   6   1   6       2   1
   7   2   1       3   1
   8   2   2       2   2
   9   2   3       1   1
  10   2   4       3   2
  11   2   5       1   2
  12   2   6       2   1

BLOCK FACTORS
  B1  BLOCKS     B2  PLOTS

TREATMENT FACTORS
  T1  N          T2  P

   $C THE 'NEXT' DIRECTIVE ENDS ONE JOB AND STARTS ANOTHER
$NEXT


   $C EXAMPLE 1A - AS 1 BUT WITH NO COMMENTS, HEADINGS ETC
$FACT BLOCKS 2 PLOTS 6 N 3 P 2
$BSTR BLOCKS/PLOTS $TSTR N*P
$ASSIGN $RANDOMIZE     $C ... AND PRINT
$END
```

```
   $C EXAMPLE 2 - SPLIT-PLOT DESIGN WITH DIFFERENT FORMS OF OUTPUT
$FACT BLOCKS 2 MAIN 2 SUB 4  VARIETY 2 N 2 P 2
$BSTR BLOCKS/MAIN/SUB  STSTR VARIETY*P*N
$ASSIGN $RANDOMIZE $PRINT $

UNIT  B1  B2  B3      T1  T2  T3
   1   1   1   1       1   1   1
   2   1   1   2       1   2   2
   3   1   1   3       1   2   1
   4   1   1   4       1   1   2
   5   1   2   1       2   1   1
   6   1   2   2       2   2   2
   7   1   2   3       2   1   2
   8   1   2   4       2   2   1
   9   2   1   1       2   1   1
  10   2   1   2       2   2   2
  11   2   1   3       2   2   1
  12   2   1   4       2   1   2
  13   2   2   1       1   2   2
  14   2   2   2       1   1   1
  15   2   2   3       1   1   2
  16   2   2   4       1   2   1

BLOCK FACTORS
  B1  BLOCKS     B2  MAIN       B3  SUB

TREATMENT FACTORS
  T1  VARIETY    T2  P          T3  N

   $C 'VARIETY' HAS NO FACTOR NAME AND 8-CHARACTER LEVEL NAMES
      'N' HAS A FACTOR NAME (N) AND 2-CHARACTER LEVEL NAMES
      'P' HAS A FACTOR NAME (P); LEVEL NAMES ARE REPLACED BY 1 AND 2
$NAMES VARIETY 8  'ZEPHYR' 'MIDAS'      N 'N' 2 '10' '20'  P 'P'
   $C THE %PRNAMES OPTION REPLACES FORMAL LEVEL NUMBERS BY LEVEL NAMES
      THE %NONUMBER OPTION CAUSES PLOT NUMBERS TO BE SUPPRESSED
$PRINT %PRNAMES %NONUMBER $

    BLOCKS    TREATMENTS
    111       ZEPHYR  P1N10
    112       ZEPHYR  P2N20
    113       ZEPHYR  P2N10
    114       ZEPHYR  P1N20
    121       MIDAS   P1N10
    122       MIDAS   P2N20
    123       MIDAS   P1N20
    124       MIDAS   P2N10
    211       MIDAS   P1N10
    212       MIDAS   P2N20
    213       MIDAS   P2N10
    214       MIDAS   P1N20
    221       ZEPHYR  P2N20
    222       ZEPHYR  P1N10
    223       ZEPHYR  P1N20
    224       ZEPHYR  P2N10

BLOCK FACTORS
  B1  BLOCKS     B2  MAIN       B3  SUB

TREATMENT FACTORS
  T1  VARIETY    T2  P          T3  N

$PAGE $
```

```
   $C THE EXPERIMENTAL PLAN CAN BE PRINTED AS A 2-DIMENSIONAL FIELD PLAN
      THE LAYOUT DIRECTIVE GIVES THE NUMBER OF ROWS AND COLUMNS FOR THE
       LEVELS OF EACH BLOCK FACTOR
$LAYOUT 4 4  BLOCKS 4 2  MAIN 2 2  SUB 1 1
   $C BASIC FORM OF MAP
$MAP $

0000000000000000000000000000000000000000000000000
0          3          1          3          0
0 111      3 122      1 211      3 222      0
0        1 3        2 1        9 3       10 0
0333333333333333333333133333333333333333333330
0          3          1          3          0
0 121      3 112      1 221      3 212      0
0        3 3        4 1       11 3       12 0
0222222222222222222221222222222222222222220
0          3          1          3          0
0 211      3 222      1 122      3 111      0
0        5 3        6 1       13 3       14 0
0333333333333333333333133333333333333333333330
0          3          1          3          0
0 212      3 221      1 112      3 121      0
0        7 3        8 1       15 3       16 0
0000000000000000000000000000000000000000000000000

BLOCK FACTORS
  B1  BLOCKS     B2  MAIN       B3  SUB

TREATMENT FACTORS
  T1  VARIETY    T2  P          T3  N

   $C AN ALTERNATIVE - SUPPRESS PRINTING OF MARGINS AND CAPTIONS
      ALSO PRINT TREATMENTS ONLY WHEN FACTOR N IS AT LEVEL 1
      AND RENUMBER THE PLOTS
$NUMBER %BYROWS
$RESTRICT N 1 1
$MAP %COMPRESS %NOCAPTIONS $

 111                 211
        1          2          3          4

 121                 221
        5          6          7          8

 211                            111
        9         10         11         12

            221                 121
       13         14         15         16


$END
```

```
   SC EXAMPLE 3
   AUTOMATIC DESIGN KEY: HALF-REPLICATE 2**6 IN 4 BLOCKS OF 8
$FACT BLOCKS 4 PLOTS 8   A 2  B 2  C 2  D 2  E 2  F 2
$BSTR BLOCKS/PLOTS  $TSTR A*B*C*D*E*F
   SC CONVERT BLOCK FACTORS INTO PSEUDO FACTORS
      AND SELECT STANDARD DESIGN KEY
$P $DESKEY %FRAC 1 1
   $C IN THE LAYOUT DIRECTIVE THE INNERMOST FACTOR CAN BE OMITTED IF IT
      IS 1*1
$LAYOUT 8 4  BLOCKS 8 1
$MAP S

000000000000000000000000000000000000000000000000
0          1          1          1          0
0 111111   1 212111   1 221111   1 122111   0
0        1 1        9 1       17 1       25 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 221122   1 122122   1 111122   1 212122   0
0        2 1       10 1       18 1       26 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 122221   1 221221   1 212221   1 111221   0
0        3 1       11 1       19 1       27 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 212212   1 111212   1 122212   1 221212   0
0        4 1       12 1       20 1       28 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 222211   1 121211   1 112211   1 211211   0
0        5 1       13 1       21 1       29 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 112222   1 211222   1 222222   1 121222   0
0        6 1       14 1       22 1       30 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 211121   1 112121   1 121121   1 222121   0
0        7 1       15 1       23 1       31 0
022222222221222222222212222222222122222222220
0          1          1          1          0
0 121112   1 222112   1 211112   1 112112   0
0        8 1       16 1       24 1       32 0
000000000000000000000000000000000000000000000000

BLOCK FACTORS
  B1  BLOCKS     B2  PLOTS

TREATMENT FACTORS
  T1  A          T2  B          T3  C          T4  D
  T5  E          T6  F

$END
```

```
   $C EXAMPLE 4 - PRINTING UNDER RESTRICTION
      A PARTIALLY CONFOUNDED (2**3)*3 IN 12 BLOCKS OF 6 PLOTS
$FACTOR SBLOCK 3 BLOCK 4 PLOT 8 A 2 B 2 C 2 D 4
$BSTR SBLOCK/BLOCK/PLOT $TSTR A*B*C*D
   $C DENOTE D EFFECTS D1, D2, D3.  THEN RESPECTIVELY CONFOUND
      ABD1, ACD2, BCD3;  ABD., ACD., BCD.;  ABD,, ACD:, BCD.
$RESTRICT SBLOCK 1 0
SP SLEVELS $
    BLOCK PSEUDO FACTORS  2  2  2  2  2
TREATMENT PSEUDO FACTORS  2  2  2  2  2
$DESKEY %READ
1 0 0 0 0
0 1 0 0 0
1 1 0 0 1
0 1 0 1 0
1 1 1 0 0
$RESTRICT SBLOCK 2 0
$DESKEY %READ
1 0 0 0 0
0 1 0 0 0
0 1 0 0 1
1 0 0 1 0
1 1 1 0 0
$REST SBLOCK 3 0
$DESKEY %READ
1 0 0 0 0
0 1 0 0 0
1 0 0 0 1
1 1 0 1 0
1 1 1 0 0
   $C NOTE LEVEL 4 OF FACTOR D IS ALWAYS ON PLOTS 7 AND 8
$RESTRICT SBLOCK 1 3 PLOT 1 6
$RANDOMIZE
$LAYOUT 6 12  SBLOCK 6 4  BLOCK 6 1
$MAP %COMPRESS %NONUM %NOCAP $

 2122  1223  2111  2213  2112  2123  2113  2213  2121  1111  1222  1221

 1121  2121  1213  2112  2121  2221  2211  1123  1123  2122  2123  1122

 2223  2212  1112  2221  1211  1213  1223  2111  1211  1212  2112  2212

 1212  2113  2222  1111  2223  1122  1121  2122  2222  2221  1121  2223

 2211  1211  2123  1123  1222  2212  2222  1212  1112  1223  2211  1113

 1113  1122  1221  1222  1113  1111  1112  1221  2213  2113  1213  2111

$END
```

Note: This design is formed in three stages. Firstly a 2^5 design is formed which is then converted into a $2^3 4$ design. Finally one level of factor D is eliminated by restrictions placed on the plots.

```
   $C EXAMPLE 5 - DESIGN WITH MIXED TREATMENT SET FORMED BY
      RESTRICTING, AND SELECTIVE SUPPRESSION OF TREATMENT LEVEL NAMES
$FACT BLOCKS 2 PLOTS 5  SET 2 CONTROL 2 A 2 B 2
$TSTR SET/(A*B*CONTROL)  $BSTR BLOCKS/PLOTS

   $C ASSIGN THE PLOTS WITH THE CONTROL TREATMENTS
$RESTRICT PLOTS 1 1  SET 2 2  CONTROL 2 2  A 1 1  B 1 1   $ASSIGN
   $C ASSIGN THE OTHER PLOTS IN STANDARD ORDER
$RESTRICT PLOTS 2 5  SET 1 1  CONTROL 1 1  A B   $ASSIGN
$RESTRICT  PLOTS SET CONTROL
   $C SET THE RANDOM NUMBER TO FACILITATE COMPARISON WITH EXAMPLE 6
$RANNUMBER 3804 1384 1113   $RANDOM

RANDOM NUMBERS = 3804 1384 1113

$NAMES A 'A'  B 'B'  CONTROL 7 ' ' 'CONTROL' BLOCKS 6 'UPPER' 'LOWER'
   $C DEFINE THE SUPPRESSION FOR PRINTING
$SUPPRESS  SET  1 CONTROL  2 A B  $
$PRINT %PRNAMES $

UNIT    BLOCKS      TREATMENTS
   1    UPPER 1     A1B1
   2    UPPER 2     CONTROL
   3    UPPER 3     A2B1
   4    UPPER 4     A2B2
   5    UPPER 5     A1B2
   6    LOWER 1     CONTROL
   7    LOWER 2     A1B1
   8    LOWER 3     A1B2
   9    LOWER 4     A2B2
  10    LOWER 5     A2B1

BLOCK FACTORS
  B1  BLOCKS     B2  PLOTS

TREATMENT FACTORS
  T1  SET        T2  A          T3  B          T4  CONTROL

   $C FOR COMPARISON, A STANDARD PRINT
$PRINT  $

UNIT  B1  B2      T1  T2  T3  T4
   1   1   1       1   1   1   1
   2   1   2       2   1   1   2
   3   1   3       1   2   1   1
   4   1   4       1   2   2   1
   5   1   5       1   1   2   1
   6   2   1       2   1   1   2
   7   2   2       1   1   1   1
   8   2   3       1   1   2   1
   9   2   4       1   2   2   1
  10   2   5       1   2   1   1

BLOCK FACTORS
  B1  BLOCKS     B2  PLOTS

TREATMENT FACTORS
  T1  SET        T2  A          T3  B          T4  CONTROL

$END
```

```
   $C EXAMPLE 6
   SAME AS EXAMPLE 5 BUT PRODUCED BY MERGING TWO SMALLER DESIGNS
$FACT FOUR 4 ONE 1 ALLPLOTS 5 BLOCKS 2 A 2 B 2 SET 2 CONTROL 2
$NAMES A 'A'  B 'B'  CONTROL 7 ' ' 'CONTROL'  BLOCKS 6 'UPPER' 'LOWER'

   $C FIRST DESIGN CONTAINS THE CONTROL TREATMENTS
$BSTR BLOCKS/ONE  $TSTR SET/CONTROL
   $C DECLARE THE DESIGN FOR LATER REFERENCE
$DESIGN DA
$RESTRICT SET 2 2  CONTROL 2 2 $ASSIGN

   $C SECOND DESIGN CONTAINS THE FACTORIAL TREATMENTS
$BSTR BLOCKS/FOUR   $TSTR SET/(A*B)
$DESIGN DB
$REST SET 1 1  $ASSIGN

   $C FULL DESIGN
$BSTR BLOCKS/ALLPLOTS
$TSTR SET/(A*B*CONTROL)
$DESIGN DC
$MERGE DA DB
   $C SET THE RANDOM NUMBER TO FACILITATE COMPARISON WITH EXAMPLE 5
$RANNUMBER 3804 1384 1113   $RANDOM

RANDOM NUMBERS = 3804 1384 1113

$SUPPRESS SET  1 CONTROL  2 A B
$PRINT ZPRNAMES $

UNIT    BLOCKS      TREATMENTS
   1    UPPER 1     A1B1
   2    UPPER 2     CONTROL
   3    UPPER 3     A2B1
   4    UPPER 4     A2B2
   5    UPPER 5     A1B2
   6    LOWER 1     CONTROL
   7    LOWER 2     A1B1
   8    LOWER 3     A1B2
   9    LOWER 4     A2B2
  10    LOWER 5     A2B1

BLOCK FACTORS
  B1  BLOCKS      B2  ALLPLOTS

TREATMENT FACTORS
  T1  SET         T2  A           T3  B           T4  CONTROL

$END
```

```
   $C EXAMPLE 7 - ADDING SUBPLOTS TO AN ALPHA DESIGN, USING PRODUCT

   FORMING THE ALPHA DESIGN
$FACT SBLOCKS 2 BLOCKS 6 PLOTS 4 VARIETY 24
$BSTR SBLOCKS/BLOCKS/PLOTS  $TSTR VARIETY
$DESIGN DA
$ALPHA %PWH

   $C  SET UP A SUB-PLOT DESIGN
$FACT SUB 2   METHOD 2
$BSTR SUB   $TSTR METHOD
$DESIGN DB  $ASSIGN

   $C FORM REQUIRED DESIGN BY SPLITTING EACH PLOT OF DESIGN DA
$BSTR SBLOCKS/BLOCKS/PLOTS/SUB   $TSTR VARIETY*METHOD
$DESIGN
$PRODUCT DA DB

   $C RANDOMIZE AND RANDOMIZE TREATMENTS APPLIED TO ALPHA DESIGN
$RANDOMIZE $RANTREAT VARIETY
$NAMES METHOD 2 'A' 'B'
$LAYOUT 8 12  SBLOCKS 8 6  BLOCKS 8 1  PLOTS 2 1
$MAP %PRNAMES %COMPRESS %NONUMBER $

 13A   21B   17B   11B   14A    1B   20B   19B   15A   23A    8A   16B

 13B   21A   17A   11A   14B    1A   20A   19A   13B   23B    8B   16A

 23B   12B    2A   24A   22A   20A   21B   18A   14B    3A   10B   12A

 23A   12A    2B   24B   22B   20B   21A   18B   14A    3B   10A   12B

 16B    8A    4B   10A    6A    5A   17A   15A   11B    7A   22A    6A

 16A    8B    4A   10B    6B    5B   17B   15B   11A    7B   22B    6B

  9B    3B   18A    7A   15A   19A    9A   24B    4A    2B    1B    5A

  9A    3A   18B    7B   15B   19B    9B   24A    4B    2A    1A    5B


TREATMENT FACTORS
  T1  VARIETY    T2  METHOD

$END
```

```
   $C EXAMPLE 8 - ADDING A TREATMENT FACTOR USING 'JOIN'
$FACT SUPER 3  BLOCKS 2  PLOTS 6  N 3  P 2
$TSTR N*P $BSTR SUPER /BLOCKS/PLOTS
$DESIGN DA
$ASSIGN
   $C SECOND DESIGN
$FACT K 2  $TSTR K $DESIGN DB
   $C SECOND DESIGN IS HELD ON A SECONDARY INPUT STREAM (11)
      NOTE THE ECHO IS UNAFFECTED
$INPUT 11
$ASSIGN %READ
2 1 1 2 1 2
1 2 2 1 2 1
1 2 2 1 1 2
2 1 1 2 2 1
1 2 1 2 2 1
2 1 2 1 1 2
$INPUT 5
   $C DEFINE THE THIRD DESIGN AND ASSIGN IT BY JOINING THE OTHER TWO
$TSTR N*P*K $DESIGN
$JOIN  DA DB
$PRINT %NOCAP  $

   1   1   1   1       1   1   2
   2   1   1   2       1   2   1
   3   1   1   3       2   1   1
   4   1   1   4       2   2   2
   5   1   1   5       3   1   1
   6   1   1   6       3   2   2
   7   1   2   1       1   1   1
   8   1   2   2       1   2   2
   9   1   2   3       2   1   2
  10   1   2   4       2   2   1
  11   1   2   5       3   1   2
  12   1   2   6       3   2   1
  13   2   1   1       1   1   1
  14   2   1   2       1   2   2
  15   2   1   3       2   1   2
  16   2   1   4       2   2   1
  17   2   1   5       3   1   1
  18   2   1   6       3   2   2
  19   2   2   1       1   1   2
  20   2   2   2       1   2   1
  21   2   2   3       2   1   1
  22   2   2   4       2   2   2
  23   2   2   5       3   1   2
  24   2   2   6       3   2   1
  25   3   1   1       1   1   1
  26   3   1   2       1   2   2
  27   3   1   3       2   1   1
  28   3   1   4       2   2   2
  29   3   1   5       3   1   2
  30   3   1   6       3   2   1
  31   3   2   1       1   1   2
  32   3   2   2       1   2   1
  33   3   2   3       2   1   2
  34   3   2   4       2   2   1
  35   3   2   5       3   1   1
  36   3   2   6       3   2   2

$END
```

```
   $C EXAMPLE 9
   AS 8 BUT DECLARING ALL STRUCTURES AT THE START OF
   THE JOB AND TRANSFERRING CONTROL AMONG THE DESIGNS USING THE
   'SETDESIGN' DIRECTIVE
$FACT SUPER 3 PLOTS 6 BLOCKS 2 N 3 P 2 K 2

   $C A MISTAKE
$BSTR SUPER/BLOCKS/POTS
* FAULT  9 IN BSTR - INVALID FACTOR
  SKIPPING TO NEXT DIRECTIVE
   $C CORRECTING IT
$BSTR SUPER/BLOCKS/PLOTS

$TSTR TA N*P  TB K  TC N*P*K
$DESIGN DA %TREATMENT TA
$DESIGN DB %TREATMENT TB
$DESIGN DC          $C TREATMENT TC IS ASSUMED

$SETDESIGN DA  $ASSIGN
$SETDESIGN DB
   $C SECONDARY INPUT MUST BE REWOUND
$REWIND 11 $INPUT 11
$ASSIGN %READ
2 1 1 2 1 2
1 2 2 1 2 1
1 2 2 1 1 2
2 1 1 2 2 1
1 2 1 2 2 1
2 1 2 1 1 2
$INPUT 5
   $C ... AND JOIN THEM
$SETDES DC  $JOIN DA DB
   $C OUTPUT ON SECONDARY OUTPUT STREAM. NOTE ECHO AND FAULT MESSAGES
      ARE STILL ON STREAM 6
$OUTPUT 12
$PRINT %NOCAP  $
$OUTPUT 6
$END
```

```
   $C EXAMPLE 10 - RESTRICTED RANDOMIZATION
      2**6 IN BLOCKS OF EIGHT;  ADE, BDEF, CEF CONFOUNDED WITH BLOCKS
$FACT ROWS 8 COLS 8 BLOCKS 8 PLOTS 8  A 2 B 2 C 2 D 2 E 2 F 2
$BSTR BLOCKS/PLOTS    $TSTR A*B*C*D*E*F
$P  $DESKEY %READ 1 1 1 1 0 0
                  0 1 1 0 1 0
                  1 0 1 0 0 1
                  1 1 0 1 0 0
                  1 1 1 0 1 0
                  0 1 1 0 0 1
$RAND %RRAN PLOTS 1
$LAYOUT 8 8  BLOCKS 1 8  PLOTS 1 1
$MAP %COMPRESS %NOCAP %NONUM  $

 211211  112112  121111  122221  221122  111222  222212  212121

 122112  212212  121222  222121  211122  111111  221211  112221

 122222  111221  212122  112111  221121  211212  222211  121112

 212211  122111  111112  121221  222122  221212  112222  211121

 211112  212222  111121  222111  112211  122122  121212  221221

 212111  222222  112122  221112  121121  122211  111212  211221

 211111  221222  112212  222112  122121  111122  121211  212221

 122212  222221  221111  212112  121122  111211  211222  112121


   $C QUASI LATIN SQUARE WITH THE SAME TREATMENT STRUCTURE
      CONFOUND ADE, BDEF, CEF WITH ROWS AND ADF, BDE, CDEF WITH COLS
$BSTR ROWS*COLS $DESIGN
   $C NOTE THE OLD DESIGN IS OVERWRITTEN
$P  $DESKEY %READ 1 1 1 1 0 0
                  0 1 1 0 1 0
                  1 0 1 0 0 1
                  1 1 0 1 0 0
                  1 1 1 0 1 0
                  0 1 1 0 0 1
$RAND %RRAN ROWS 1 COLS 1
$LAYOUT 8 8  ROWS 1 8  COLS 8 1
$MAP %COMPRESS  %NOCAP %NONUM $

 112111  211212  221121  111221  122222  222211  121112  212122

 221211  122112  112221  222121  211122  111111  212212  121222

 111222  212121  222212  112112  121111  221122  122221  211211

 122212  221111  211222  121122  112121  212112  111211  222221

 211112  112211  122122  212222  221221  121212  222111  111121

 212221  111122  121211  211111  222112  122121  221222  112212

 222122  121221  111112  221212  212211  112222  211121  122111

 121121  222222  212111  122211  111212  211221  112122  221112


$END
```

Chapter 10

PROGRAMMING ASPECTS OF DSIGNX

10.1 Summary

The construction of the DSIGNX program is described and related to the facilities described in Chapter 9. In §10.2 the main features of the program are outlined with emphasis on the relationship between the data structures, program organisation and the individual routines.

Data structures and program directories are described in §10.3 and the program structure in §10.4. The routines in DSIGNX fall into a few natural groups according to their function; a short description of the routines in each group is given in §10.5. In §10.6 selected algorithms from the program are described. These algorithms are usually more primitive than those in the program but are selected and modified to highlight the basic methods adopted.

10.2 Introduction

The DSIGNX program is controlled by a sequence of directives whose effects may be modified by options; the structure of the program reflects this process. The main controlling routine DSCONT determines the next directive and transfers control to routines associated with the directive which process instructions and pass control to further routines for execution. These routines also call upon others for executing basic tasks such as selecting initial blocks from a library. The program structure is therefore essentially hierarchical with four strata; it is described in more detail in §10.4.

Data structures are held within a large one-dimensional array in a common area (DSIGNDX). Storage and retrieval of information in these structures is controlled by directories also held within DSIGNDX.

Pointers are held in another common area (POINT). (All common areas used in DSIGNX are labelled). Separate directories are held for each type of object: factor, block structure, treatment structure, design and heading. For each object directories hold information on where it is stored etc. together with pointers to other directories for information on related objects (see Fig. 10.1). For example, retrieving information on a particular factor in the current design can involve linking from the design directory to the block structure directory and thence to the factor directory.

Information required by several routines is normally held in a common area; there are nine areas in all and they are described in detail in §10.3. Each routine has access to common areas if necessary, but in the more basic routines information is normally transferred through argument strings.

Directives may be grouped together by function; similarly, main routines relating to directives may be grouped. Thus routine QASSI, QALPH, QCYCL and QDESK, which control processing of directives $ASSIGN, $ALPHA, $CYCLIC and $DESKEY respectively, all have essentially similar roles and structure. This grouping of directives gives a natural division for program overlays.

DSIGNX was designed to work on small computers and to be easily transportable. The program which is written in standard FORTRAN has a modular construction and the overall size can be controlled by overlays and omitting routines. Also the sizes of the common areas can be adjusted as desired. The standard version of the program is intended for computers with a word length of 16 bits, as this is the most common with small computers. Non-standard FORTRAN instructions (e.g. for character handling) are held within special subroutines which are identified

and readily modified. Machine-dependent constants and symbols (size of arrays, word lengths, special symbols etc.) are isolated in special common areas which are readily altered by suitable BLOCK DATA routines. Although no tests have been carried out a useful version of the program could probably be mounted on a computer with a 64K memory and FORTRAN compiler.

10.3. The labelled COMMON areas

DSIGNX uses nine common areas to pass information between programs. The areas vary greatly in size with the smallest containing only eight words and the largest (usually) 2560 words. They also vary greatly in the number of routines which require access e.g. the area containing error information is required by most main routines whereas the three containing print information relate to at most three directives. All common areas contain space for program development.

The nine common areas are:

COMMON/DSIGNDX/ (length 2560 words) — Contains the main storage area of the program. Stores directories and data structures and some pointers. (§10.3.1)

COMMON/POINT/ (length 48 words) — Contains most principal pointers to program directories. It is complementary to COMMON/DSIGNDX/. (§10.3.1)

COMMON/RESLEV/ (length 320 words) — Contains information about the current design including pointers to block and treatment structures, the unrandomized design etc. (§10.3.2)

COMMON/ENVIRN/ — Contains information relating to

i) characters used by DSIGNX,

ii) restrictions and default values, input-output channels,

iii) current settings for input-output channels.

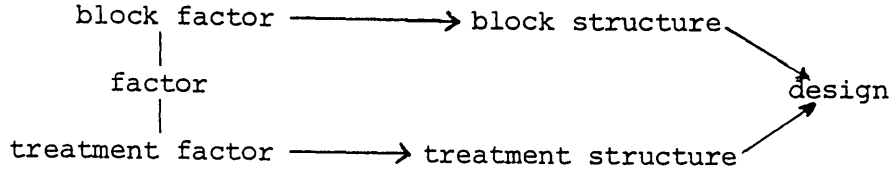
COMMON/WORKC/ (length 768 words)	Contains a general purpose work area, holding information which, in the event of an error, enables the program to be restored to its state at the start of the current directive.
COMMON/MISFIT/ (length 8 words)	Contains information for error message and recovery routines together with the current value of the pseudo-random number generator.
COMMON/DSFORM/ (length 51 words)	Contains the run-time print formats required by directives $PRINT, $MAP, and $PRHEADING.
COMMON/DISPLAY/ (length 50 words)	Contains information on the two-dimensional layout established in directive $LAYOUT and used by directives $NUMBER and $MAP.
COMMON/SUPPRI/ (length 65 words)	Contains details of the print suppression established in directive $SUPPRESS and used by directives $PRINT and $MAP.

The contents of common areas ENVIRN and POINT are displayed by directive $ENVIRONMENT, those of other areas by directive $DEBUG. Contents of ENVIRN, POINT and DSFORM which remain constant are set at the time of compilation in a BLOCK DATA program; others are initialised within DSIGNX at the start of session and reset when directive $NEXT is used.

10.3.1 DSIGNX data structures and directories

Few data structures are required in DSIGNX and some are so closely related to directories that the two topics are discussed together.

The relationship between objects (data structures), factor, block structure, treatment structure and design may be represented schematically as follows:

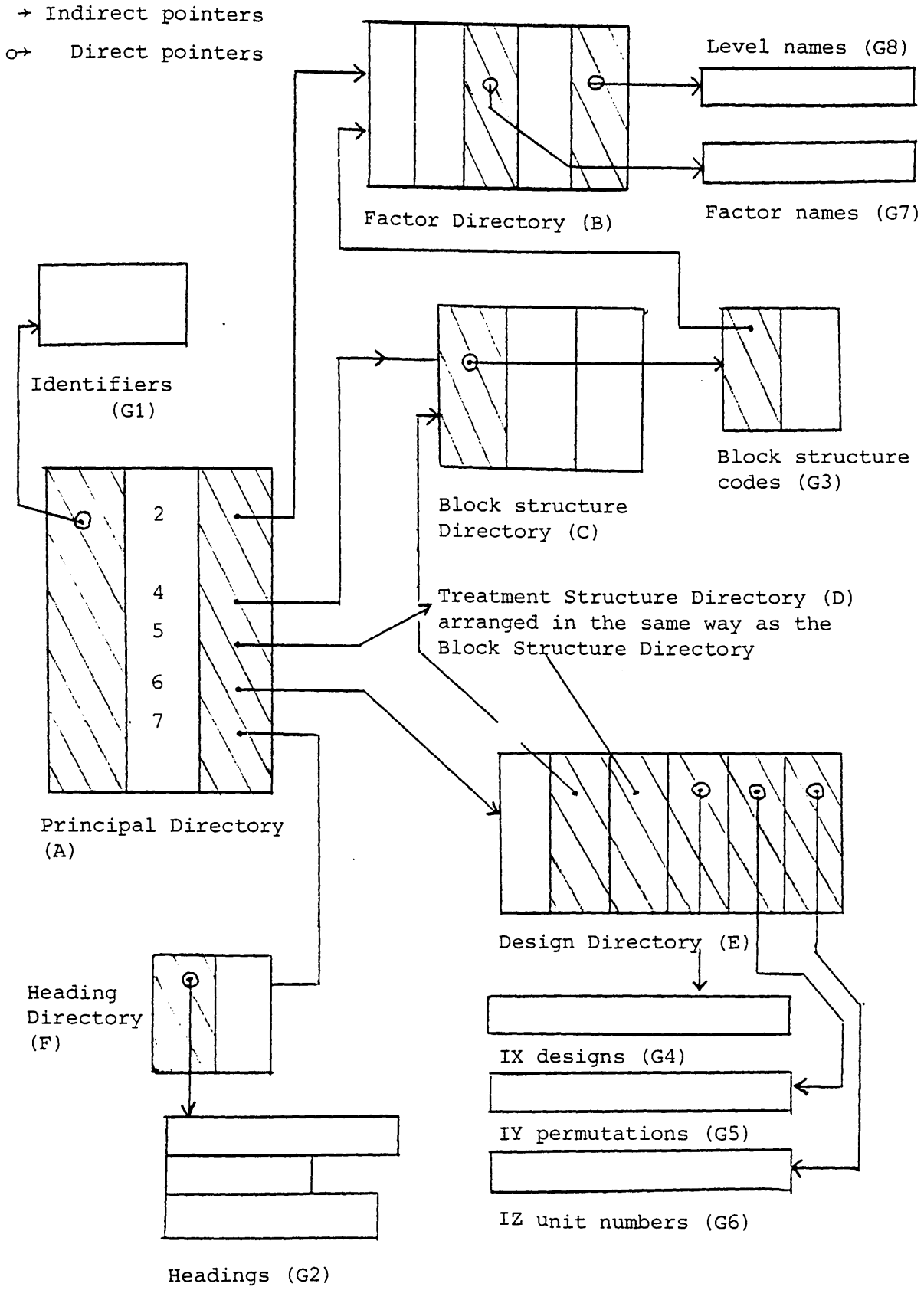


When manipulating a design, knowledge is required not only of the location of the design but also the block and treatment factors involved. This problem is overcome in DSIGNX by using a design directory which holds pointers to designs and to other directories which themselves hold pointers to yet other directories, all held in the common area DSIGNDX. The contents of this area are illustrated in Fig. 10.1 and details are given in Appendix A10.1.

Pointers, which are shown by shaded boxes in Fig. 10.1, may act directly or indirectly. When working directly they hold the location in DSIGNDX where an integer is held or where a vector starts. (Block structure formulae, for example, though represented by a compact box in Fig. 10.1 are commonly declared at separate stages in the program and are separated by objects declared in the interim.) When working indirectly the pointers hold an offset value from a location determined by a pointer in common area POINT, e.g. the start location of the factor directory is held in POINT and references to factors in DSIGNDX are relative to this location.

Figure 10.1 Arrangement of common area DSIGNDX

(See Appendix A10.1 for details)



The contents of common area POINT are summarised in Table 10.1. The names of pointers are formed by prefixing the two letters at the left of each row by the letter at the head of each column. Initial values for each variable are given in the body of the table.

directory		Type of pointer I	L	J	M	N	K
principal	ID	0	3	-	40	0	-
factor	TF	IID+LID*MID	5	-	24	0	-
b.structure	BS	ITF+LTF*MTF	3	2	5	0	0
t.structure	TS	IBS+LBS*MBS	3	2	5	0	0
design	DS	ITS+LTS*MTS	6	-	5	0	0
heading	HE	IDS+LDS*MDS	2	-	10	0	-

Table 10.1: Pointers held within common area POINT and their initial values (see text for details).

The rows of Table 10.1 refer to one of the directories shown in Fig. 10.1 and the columns indicate the type of pointer as follows:

I: location of start of directory

L: number of items (columns) per entry

J: number of items (columns) in ancillary directory

M: maximum number of entries

N: current number of entries

K: location of current structure.

The contents of columns I,J,L and M are held constant throughout the program but colums N and K are updated.

10.3.2 The current design directory

Common area RESLEV holds information required for management of the current design (§9.4). Some information is available in the main

directory DSIGNDX but is copied for convenience. The contents are:

1) Pointers to block/treatment structure directories (KSB, KST);

2) The block factor nesting matrix, LBIN(16) (§1.3), and dimension of the block structure (LDIM);

3) Pointers to the factor directory (LOCBF(16), LOCTF(16));

4) Pointers to locations of the treatment, randomization and unit number lists (IDX, IDY, IPN);

5) The working vector LVEC (3,32) used to control multi-dimensional table operations in routine NEXLEV (§10.6.2);

6) Information relating to basic, restricted and effective block/treatment structures (§9.4) summarised in the table:

	Basic	Restricted	Effective
a)	NBLF, NTRF	MBLF, MTRF	IPSDB, IPSDT
b)	LEVBFO(16), LEVTFO(16)	LEVBF(16), LEVTF(16)	MPB(16), MPT(16)
c)	-	-	MBW(16), MTW(16)
d)	NUN	NPL	NPL
e)	LBB, LBT	-	-

where

a) are the number of block and treatment factors;

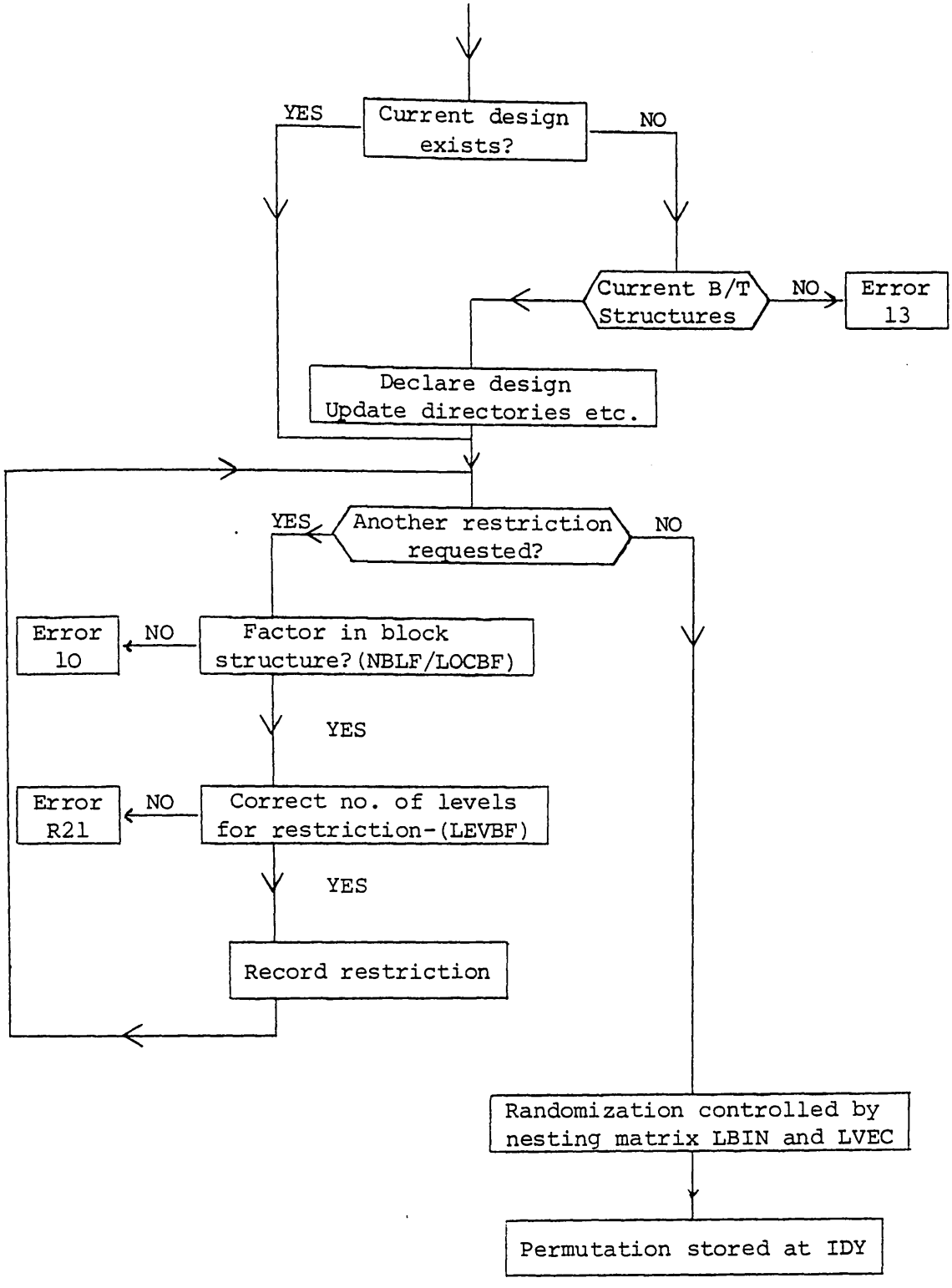
b) are vectors of numbers of factor levels;

c) are vectors linking (restricted) factors to pseudo-factors;

d) are the number of (restricted) units;

e) are locations in the lexicographical lists of the unit and treatment corresponding to the lowest factor levels allowed by current restrictions (§10.6.2).

Example: The following (simplified) flow diagram outlines how information held in DSIGNDX, POINT and RESLEV is used in directive $RANDOMIZE:


Current design exists?
YES
NO
Current B/T Structures
NO
Error 13
Declare design Update directories etc.
Another restriction requested?
YES
NO
Factor in block structure? (NBLF/LOCBF)
NO
Error 10
YES
Correct no. of levels for restriction-(LEVBF)
NO
Error R21
YES
Record restriction
Randomization controlled by nesting matrix LBIN and LVEC
Permutation stored at IDY

10.4 Program control in DSIGNX

Routines in DSIGNX are arranged mainly in an hierarchical manner with four strata:

1) main program/Subroutine DSCONT;

2) directive control routines;

3) executive routines;

4) basic routines.

The arrangement is represented schematically in Fig. 10.2. Names of directive control routines are abbreviated names of directives prefixed by Q . Most basic routines are called by several execution routines and, for clarity, are not shown but listed under six headings below the figure.

Subroutine DSCONT is in essence a main program and can usually be converted to such by removal of the SUBROUTINE statement and replacing the RETURN statement by END. Some computers however are specific in the form required for a main program and one has to be written to perform essential functions and call DSCONT. On the Edinburgh Multi-Access System (EMAS), for example, there are two distinct main programs. One, a short FORTRAN program, initialises the program's 'prompt' message but requires the user to write his own job control instructions; the other, a more extensive program written in the language IMP, writes the job control language on behalf of the user. Other main programs which allow, say, the job control instructions to be established by a conversational process have been written.

The job of DSCONT is to cause various items, such as the seed for the random number generator to be initialised, to read directive names and to call the appropriate directive control routines. When control is returned to DSCONT an error indicator is checked, the error message

Fig. 10.2 Arrangements of routines in DSIGNX

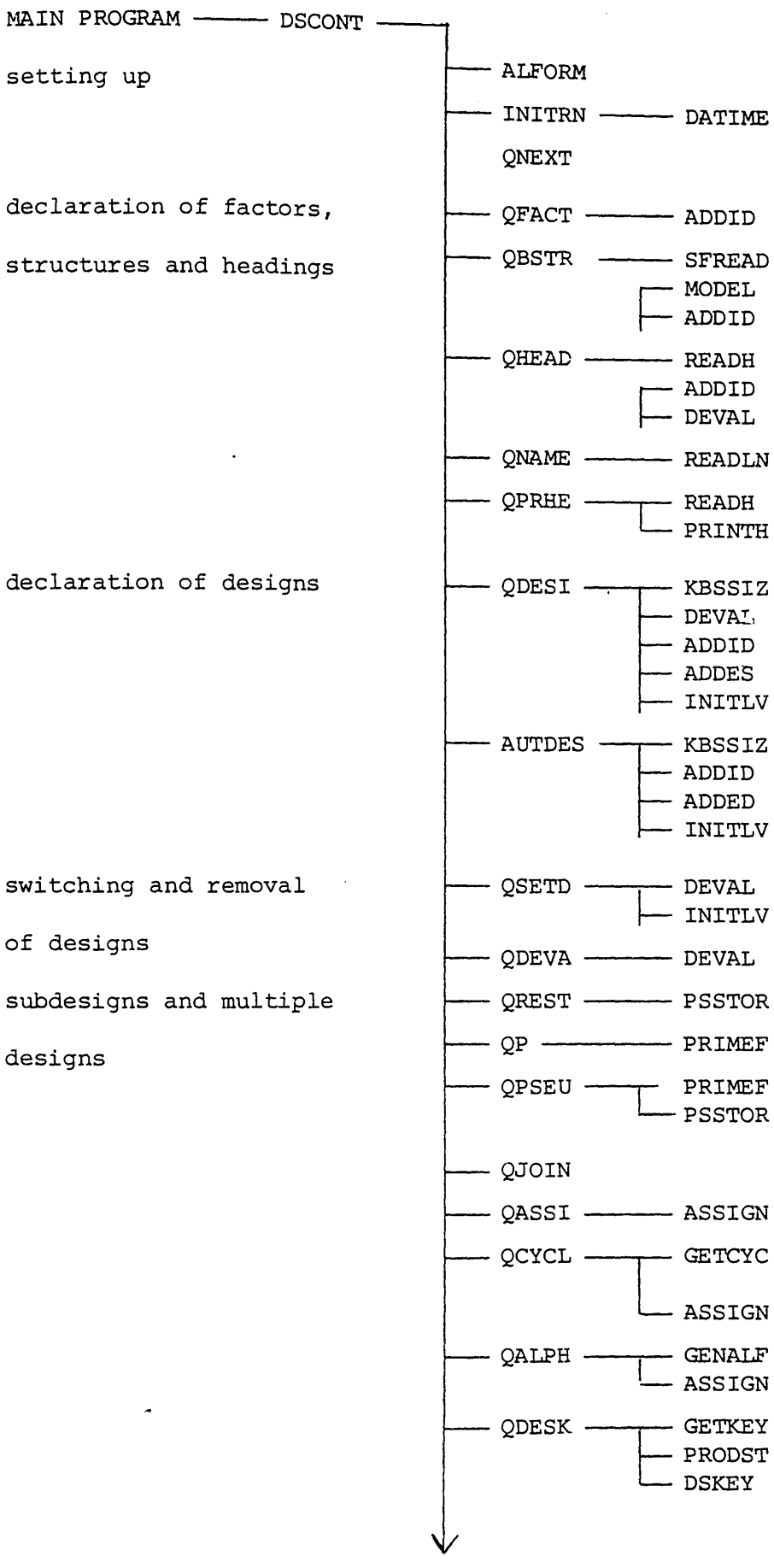
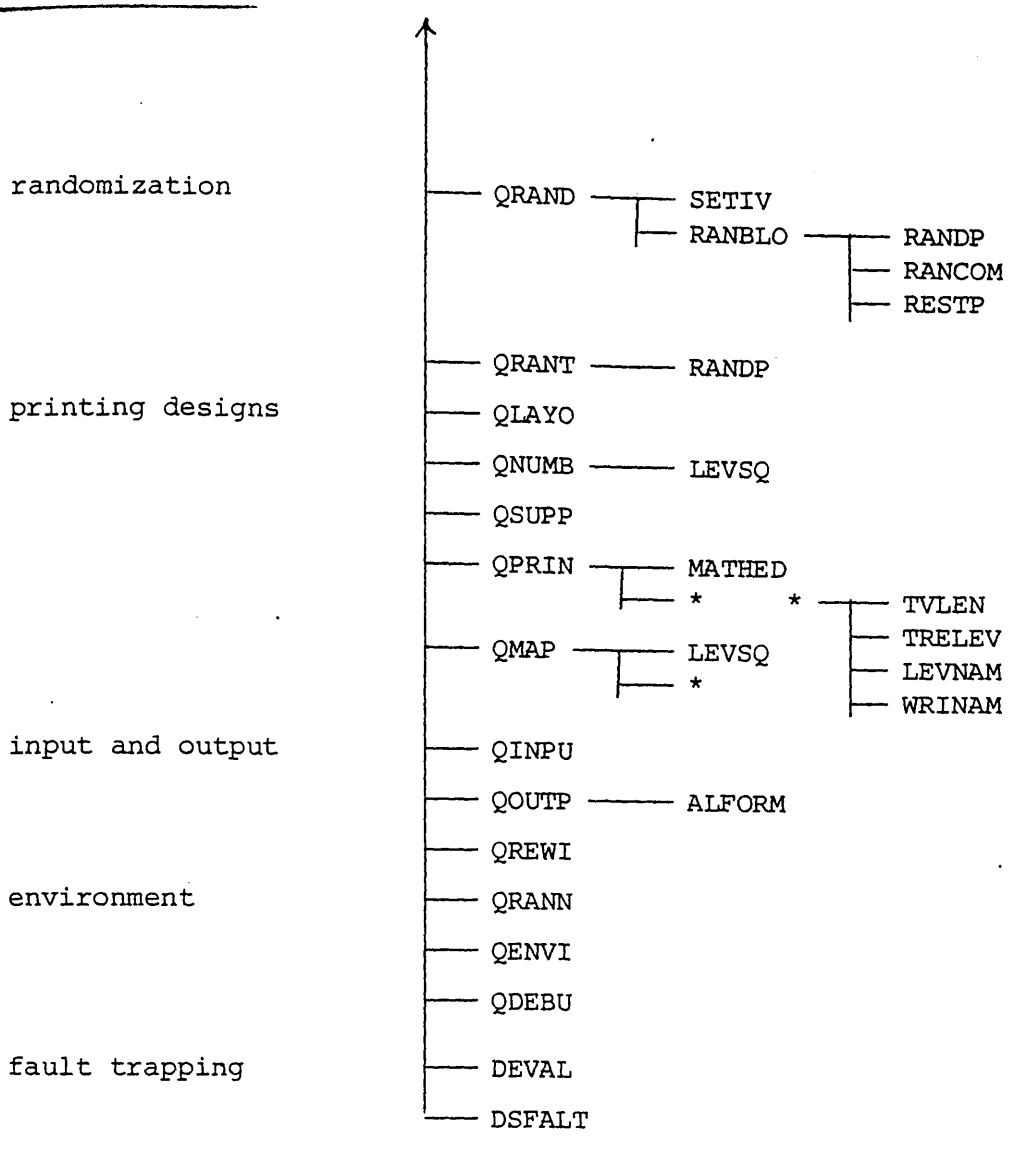


Fig. 10.2 (contd)



The following basic routines, approximately grouped by function, are called by several others and are not shown in the above diagram.

1) direct-access file input: DAREAD
2) free-format input-output: (14 routines)
3) character handling: FINDIR, GETID, GETOP, TOSP, ZNOZAI
4) directory search: DSFAC, FCNAME, INLIST
5) bit manipulation: ICNT, IEOR, JTHBIT
6) multi-dimensional arrays: NEXLEV, NOLEV, SETLEV.

Directives processed within DSCONT:

£C, $ECHO, $END, $LABEL, $PAGE, $RANNUMBER, $REWIND, $STEP.

and recovery routine called if necessary and the next directive read. Directive control routines work in two stages. Firstly they cause input instructions to be read as far as the next symbol $ and then interpreted and checked for errors. Secondly, if no error occurs, they cause the instructions to be obeyed. The first stage is carried out within the directive control routine with the aid of some general-purpose routines. The second stage is, except for a few directives such as $ECHO and $REWIND, performed through calling an executive routine.

Executive routines vary greatly in their nature and purpose. For directives $BSTRUCTURE and $TSTRUCTURE, for example, they decode structure formulae and update directories; for directive $CYCLIC they generate cyclic designs. (Some of the more important executive routines are discussed in §10.6.) Basic routines are often called from several executive routines and perform a variety of fundamental tasks such as free-format input, searching of directories, bit manipulation, reading direct-access files etc.

Subroutine DSCONT and the directive control routines are specific to DSIGNX and are affected by the syntax and method of storing the directories but the executive routines vary in specificity. Basic routines are program independent but not necessarily machine independent; the machine dependent routines, apart from routine DATIME for accessing the computer clock, are those for manipulating characters or reading direct-access files.

10.5 Outline of routines

The main features of the routines are described. For convenience the routines are grouped into declaration, allocation, randomization, output and others according to how the associated directive is classified in Chapter 9.

10.5.1 Declaration routines

Declaration directives fall into two groups, those which modify directories shown in Fig. 10.1 and detailed in Appendix A10.1:

$FACTOR, $BSTRUCTURE, $TSTRUCTURE, $DESIGN, $HEADING, $DEVALUE

and those which modify the current design directory (§10.3.2):

$SETDESIGN, $RESTRICT, $P, $PSEUDO, ($LEVELS) .

When any of the objects, factor, block structure, treatment structure, design or heading is declared its identifier is checked, using routine ADDID, against the principal directory to see whether it is a new entry and, if not, that the previous entry is of the same type. If acceptable, the principal directory is updated by recording the identifier, its type and setting a pointer to the appropriate directory.

For a factor the number of levels is entered in the first column of the factor directory. For a heading the character string is stored within free area of DSIGNDX and its starting location and length entered in the heading directory. Block and treatment structures are treated similarly upon declaration but use different directories. The formula is read, decoded and checked; on successful decoding the number of factors and the dimension (i.e. the degree of crossing) are stored in the structure directory. Also stored is the nesting matrix (§1.3) and a pointer to a sub-directory containing pointers to the factor directory. The last structure declared in a statement is deemed to be current. Directive $DESIGN causes a new entry in the design directory with pointers to the current block and treatment structures. The number of units in the (unrestricted) design is calculated and areas of free space in DSIGNDX are allocated for the design, permutation and plot numbers. The number of units and start locations are stored in

the design directory. The directive also causes a new current design to be declared and details copied to the current design directory (§10.3.2). (This last step is also performed by directive $SETDESIGN.)

Designs and headings can be deleted by directive $DEVALUE which removes the reference from the associated directory and recovers free space by repacking the main storage area. Block and treatment structures cannot be deleted but an unnamed block structure, treatment structure or design which ceases to be current is no longer accessible so it is deleted. Of the other declaration directives, $RESTRICT, $PSEUDO, and $P affect only the current design directory and directive $LEVELS causes the effective block and treatment structures to be printed (i.e. arrays denoted MPB and MPT in §10.3.2).

10.5.2 Allocation routines

Allocation (or design construction) routines for directives $DESKEY, $ASSIGN, $CYCLIC and $ALPHA are essentially simple generators which are given greater versatility by the use of restrictions and pseudo-factors. In each case the design is treated as a multi-dimensional array with multi-variate entries which is simulated by a one-dimensional array (§10.5.5) denoted IX in Fig. 10.1; restrictions on block factor levels determine sub-arrays. The routines process units in lexicographical order (as determined by current restrictions), determine the treatment to be allocated and then store it in compact form at the appropriate location of array IX.

Routines $MERGE, $PRODUCT and $JOIN also process the units in lexicographical order but determine the treatments from those allocated to the appropriate units of the designs to be combined.

10.5.3. Randomization routines

Pseudo-random numbers are formed by the multiplicative

congruential generator given by Knuth (1969, p. 88); the ith number generated x_i is given by

$$x_i = ax_{i-1} + c \pmod{m}$$

where $a = 2^{23} + 2^{14} + 2^2 + 1$ (= 8404997), $c = 1$ and $m = 2^{35}$. The generator uses a word length of two bytes and values x_i are held as the composite of three integers, two of which are less than 4096 and one (the second) less than 2048.

The array IY (Fig. 10.1) holds a permutation of the numbers 1 ... n = the number of units in the unrestricted design. This permutation determines the relationship between the basic design and the experimental plan. The ith unit corresponds to the ith cell and the treatment allocated to the unit is given by IX(IY(I)) where IX is the array holding the basic design. Initially the ith cell of IY contains the value i ; randomization causes the replacement of the contents of IY by a random permutation.

The default randomization is controlled by the nesting matrix (§1.3). The levels of each block factor are randomized for each combination of levels of the factors containing it. The program processes the factors in the order they appear in the block structure. If restricted randomization of the Grundy and Healy (1950) type is required the number of (restricted) levels of the associated factor is checked and restricted randomization applied to the levels as described in §7.4.

For treatments the formal levels of specified factors are randomized after the design is constructed.

10.5.4 Routines for the output of the experimental plan

Directives $PRINT and $MAP share many basic routines, their differences being confined mainly to the directive control routines.

For directive $PRINT units are processed in lexicographical order (as determined by any restrictions), the associated treatments determined from the design 'array' IX (Fig. 10.1) and the permutation 'array' IY and printed according to the options requested. The unit number is determined from array IZ (Fig. 10.1). For directive $MAP there must exist a two-dimensional grid defined in directive $LAYOUT and held in common area DSLAY. For each cell of the grid the corresponding unit is determined and the treatment and plot number obtained as for directive $PRINT. The total grid is then printed under the control of options %ROW and %COLUMN.

Directive $NUMBER causes array IZ (Fig. 10.1) to be altered; when numbering follows rows or columns of the two-dimensional layout routine LEVSQ (Fig. 10.2) is used to determine the location in IZ associated with each cell of the layout.

Both print routines use four routines (denoted TVLEN, TRELEV, LEVNAM and WRINAM in Fig. 10.2) to construct treatment names from level names. The routines use information held in common area SUPPRI to determine which parts of a name are to be suppressed.

10.5.5 Other routines

Routines associated with directives listed in §9.9 are mostly straightforward and need not be described in detail. For example, directive $OUTPUT has three tasks

i) to change the current stream number

ii) to change the length of output records

iii) to reset the run-time output format statements.

The first two tasks are trivial and the third is performed by ALFORM (Fig. 10.2) at the start of each run.

The basic routines listed beneath Fig. 10.2 are, for the most part, both program and machine independent but, except for routines SETLEV, NEXLEV and NOLEV, do not require description here. All three exceptions are concerned with simulating multi-dimensional tables in a one-dimensional vector and are used in the allocation, randomization and output procedures. NEXLEV is used to process cells of a subtable in lexicographical order and optionally to indicate which position in the lexicographical list corresponds to a given cell. SETLEV is an initialisation routine and NOLEV indicates which cell corresponds to a given position in the list. NEXLEV is described in more detail in §10.6.2 and a FORTRAN routine based on SETLEV and NEXLEV is given in Appendix A10.2.

10.6 Some useful algorithms

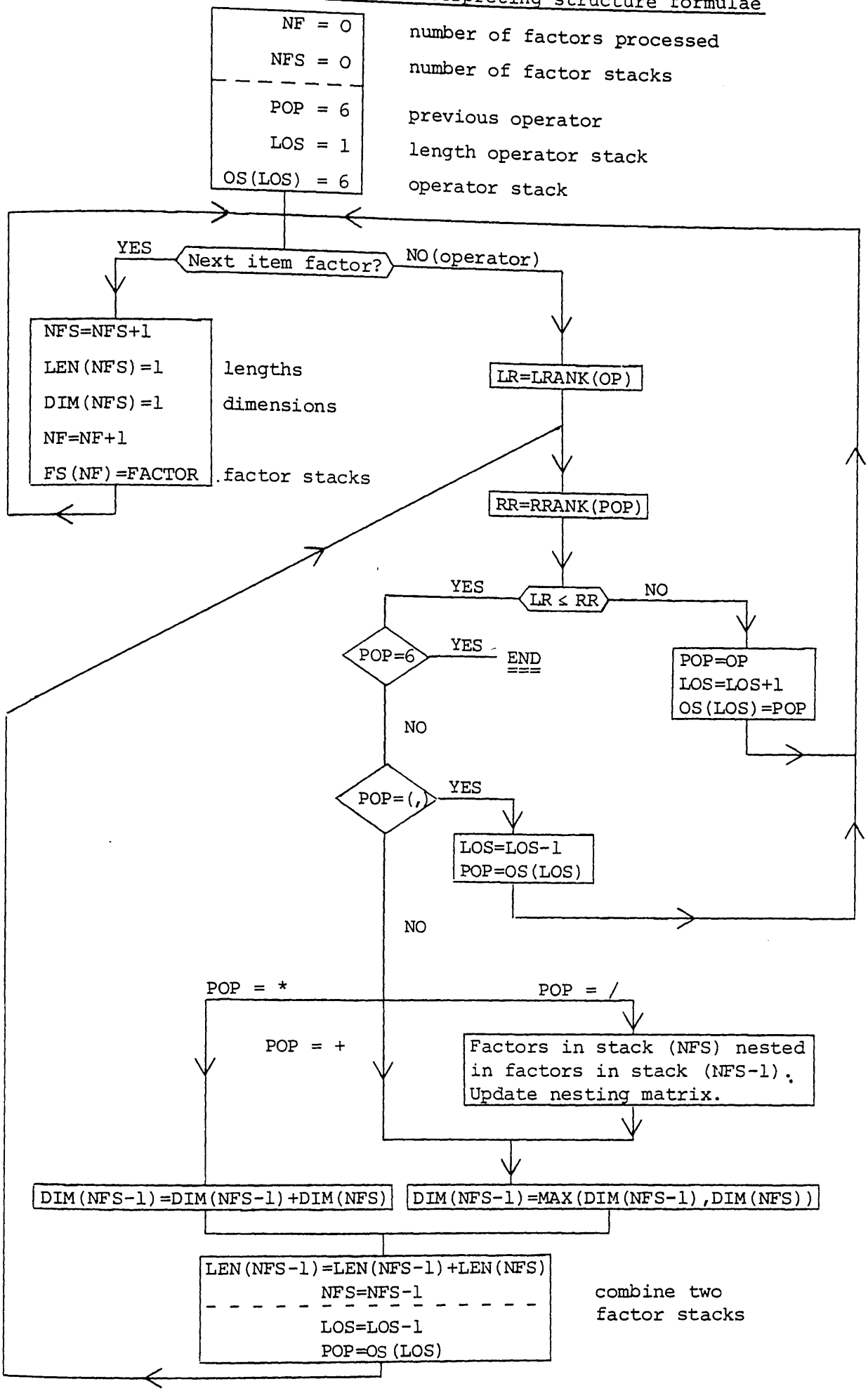
Algorithms closely related to some used in DSIGNX are presented. For the sake of clarity, those presented here are simpler than the corresponding routines in the program but the main features are similar.

10.6.1 Interpretation of block and treatment structure formulae

Useful algorithms for interpreting structure formulae are given by Rogers (1973) and Fowlkes and Lee (1971, p. 238). The algorithm presented here is related to the latter but is simpler and much shorter.

The interpretation of formulae is a two-stage process. The formula is read, the syntax checked and the indentifiers verified as belonging to factors, none occurring twice. The formulae are then coded and passed as an array to the second stage. The coding is as follows:

Fig. 10.3 Flow diagram for interpreting structure formulae



symbol :	/	*	+	(	)	terminator	ith factor
code :	1	2	3	4	5	6	-i

Example: The structure formula (A*B)/C is coded

4 -1 2 -2 5 1 -3 6

A simple precedence class for the operators is defined by two vectors LRANK and RRANK

Code:	1	2	3	4	5	6
LRANK:	4	3	2	5	1	0
RRANK:	4	3	2	1	5	0

The nesting matrix and dimension are then determined by a procedure which is summarised in the flow chart presented in Fig. 10.3.

10.6.2 Procedures for multi-dimensional arrays

The procedures described here are related to those described in algorithms AS1 and AS1R1 (Gower (1968, 1969)). They are used in constructing the design, randomizing and printing procedures. We show how they handle restrictions and pseudo-factors.

Consider n treatment factors with t_i levels (i=1, 2, ...n) and levels of the ith factor 0, 1 ...t_i-1 . Set $m_n = 1$ and $m_i = m_{i+1}\ t_{i+1}$ $(1 \leq i < n)$ then the position of treatment $(k_1\ k_2\ \ldots k_n)$ in the lexicographical ordering of treatments is

$$loc = \sum_{i=1}^{n} k_i m_i \qquad (0 \leq loc < \Pi t_i) \qquad 1)$$

Let restrictions on factor levels be of the form $a_i \leq k_i \leq b_i$ (i=1, 2 ... n) and define modified factors with $t_i' = b_i - a_i$ levels whose jth levels corresponds to levels $a_i + j$ of the original factors. If $(k_1', k_2' \ldots k_n')$ is a treatment expressed in levels of the modified factors then its position in the

lexicographical list is given by

$$loc = \sum_i (k_i' + a_i)m_i$$

$$= \sum_i k_i' \; m_i + t_{min} \qquad 2)$$

where t_{min} is the location in the list of treatment $(a_1, a_2 \ldots a_n)$.

Let the pth (modified) factor with t_p' levels be replaced by s pseudo-factors with u_j levels (j=1,2, ...s) such that $\prod_j u_j = t_p'$ then the treatments may be represented by a vector of the form:

$$(k_1', k_2' \ldots \ell_1, \ell_2 \ldots \ell_s \ldots k_n') \qquad 3)$$

Let $w_s = 1$ and $w_j = w_{j+1} u_{j+1}$ $(1 \leq j < s)$ then the location of treatment 3) in the lexicographical ordering of treatments is

$$loc = \sum_{i \neq p} k_i' \; m_i + (\sum_j \ell_j \; w_j)m_p + t_{min}$$

$$= \sum_{i \neq p} k_i' \; m_i + \sum_j \ell_j \; m_j' + t_{min} \qquad 4)$$

where $m_j' = w_j \; m_p$.

The expressions 1), 2) and 4) can all be written in the form

$$\underline{k}^T \; \underline{m} + a \qquad 5)$$

where $\underline{k}$ is a vector of (pseudo) factor levels, $\underline{m}$ is a vector of accumulated products (or, more generally, locations) and a is the location of the treatment with lowest levels for all factors.

In DSIGNX multidimensional arrays are handled by three vectors, K a vector of levels, L a vector holding the number of levels of the (pseudo) factors and M a vector containing accumulative products m_i and m_j' used in expression 4) above. For convenience, the three vectors for both units and treatments are held in a two-dimensional array LVEC. This array is used in routine NEXLEV (§10.5.5) to control processing of multidimensional arrays.

The control of multi-dimensional arrays requires two routines, one to initialise vectors L and M (cf. SETLEV) and one to update vector K and give the lexicographical location (cf. NEXLEV). An algorithm is given for the former; one for the latter is readily derived from the routine presented in Appendix A10.2.

Algorithm effective_structure:

Variables:

N : the number of factors

MA : vector of products for the unrestricted factors (m_i) (length N)

T : vector of level numbers for restricted factors (t_i') (length N)

SP,LB : pointers for pseudo-factor stack, output vectors

LOC : location of pseudo-factors in stack (length N)

NPF : number of pseudo-factors for each factor (length N)

TPF : total number of pseudo-factors

LP,MP : output vectors L and M (length TPF)

1) Input N, MA, T; initialise SP ← 0, NPF ← 1, LOC ← 0

Remark: a subset of factors are specified in arbitrary order. Location I of each factor in the structure formulae is to be determined.

2) Loop in_factor : read factor name and pseudo-factor list of length P ; determine I and check pseudo-factors compatible with T(I); copy pseudo-factors to stack from locations SP; NPF(I) = P LOC(I) = SP; SP = SP + P; end in_factor

3) TPF = SP + no. factors with no declared pseudo-factors; I = N; LB = TPF;

Remark: the pseudo-factors are to be copied from the stack to output arrays.

4) Loop copy_factor MP(LB) = MA(I) ;
if factor I has pseudo-factors then
use LOC(I), P = NPF(I) to copy pseudo-factors from stack to LP
(backwards from LB) updating to MP in process; LB = LB - P else
LP(LB) = T(I) ; LB = LB - 1
I=I- ; end copy_factor;

5) Output LP, MP; end

10.6.3 Allocation routines

Allocation routines $ASSIGN, $ALPHA, $CYCLIC and $DESKEY construct the design through simple generators and correspond to different functions for allocating treatments to units. As complete routines they have essentially the same programming 'overheads' (methods of generating the units, methods of storing the treatments etc.). For brevity the discussion is restricted to directive $CYCLIC. (A routine for a simple form of the design key generator is given in Appendix A10.2).

Algorithm cyclic:

Variables:

v,k,b : have the usual meaning for incomplete block designs and apply to the design as currently restricted

NTF : number of treatment factors in the effective treatment structure =1 for options %JWD, %DH; =2 for option %HW; ≥1 for option %READ

INC : increment value, a divisor of v , default value =1

S : number of blocks formed from initial blocks. Usually S = v/INC but the last initial block of a set is allowed to generate fewer blocks

NPL : number of units in the restricted design

TB : vector holding current block (length k*NTF)

TINC : temporary vector of increments (length k*NTF)

TT : vector holding current treatment (length NTF)

NIB : expected number of initial blocks

1) Obtain default k value and NTF from current design directory;
If NTF > 1 check number of levels equal; record locations of factors in treatment structure;

2) If k input, check k divides NPL; set new k, b ;
If INC input, check INC divides v ; set INC;
Calculate NIB;
Remark: initial blocks may be input or obtained from library.

3) If option is %READ then
read integer list; check length =NIB*k ;
check integers conform to unrestricted factor levels; else

check k, b, t, NIB, NTF agree with selected design;
retrieve initial blocks from library;

Remark: input list has ≤3 dimensions, each element generates s ≤ t/INC units.

4) Loop initial_block:
locate new initial block; store in TB; KF = 0;
calculate number of blocks S to be generated;

5) Loop k Loop factor: KF = KF + 1 ;
If TB(KF) allowed by restriction then TINC=INC else TINC = 0
end factor; end k ;

6) Loop block:
update location of unit LOC (routine NEXLEV);

7) Loop unit
update location of unit LOC; KF = 0;

8) Loop factor: KF = KF+1
TT(KF) = TB(KF); TB(KF) = TB(KF) + TINC(KF) mod v end factor;

9) convert TT to treatment and store in array W at LOC end unit;

10) If block ≥s end block; end initial_block;

11) Adjust W and copy to IX(Fig. 10.1); end .

10.6.4 Algorithms for $MERGE, $PRODUCT and $JOIN

The three directives for forming the current design from two others work on somewhat similar principles. For brevity, we discuss only the routine for directive $MERGE (§ 9.6.4).

Merging two designs by directive $MERGE

Let E_1, E_2 be block structure formulae; FA, FB be block factors; DA, DB be designs with block structure $E_1 \ \emptyset \ F_1 \ \theta \ E_2$ and $E_1 \ \emptyset \ F_2 \ \theta \ E_2$ respectively, where $\emptyset$ and θ denote operators * or / . DA and DB can be merged to form the (restricted) design DC if

i) the (restricted) block structure of DC is $E_1 \oslash F_3 \theta E_2$ where $F_3 = F_1 : F_2$, the union of factors F_1 and F_2;

ii) the levels of the treatment factors occurring in designs DA and DB are compatible with the levels in design DC.

Option %KEEPRANDOM allows the permutation lists (§§10.6.5; 10.3.1) to be copied yielding a randomization for factor F_3 equivalent to randomizing the first f_1 and last f_2 levels separately.

If a factor occurs in the treatment structure of design DC but not of design DA, the latter design is treated as if the factor has a single level. (DSIGNX allocates formal level 0 but it should be optional.) A similar action is taken for designs DC and DB.

Algorithm merge:

Variables

MA : main storage array for design and randomization

KE : no. of combinations of factor levels in E_1

IRA, IRB : no. of combinations of factor levels in $F_1 \theta E_2$, $F_2 \theta E_2$

IXA, IXB : locations of unrandomized designs DA, DB in MA

IYA, IYB : location of permutations for designs DA, DB in MA.

1) Check compatability of block structures of designs DA, DB and DC; locate factors for merging.

Remark: The treatment structure formulae of the three designs may be very different. Factors in designs DA, DB need to be related to factors in design DC.

2) Check compatibility of treatment structures; record locations of factors in DA and DB in formula for DC; set levels for missing factors.

Remark: For each combination of factors in E_1 copy units from DA and then from DB. If %KEEPRANDOM requested locate permutation value, update to allow for units from other designs and store.

```
3)  Loop external

4)  Loop internal_A

        next unit  DC  (routine NEXLEV);

        IXA = IXA + 1;  unpack treatment from  DA ;

        reorder factors;  pack treatment for  DC ;

5)  If %KEEPRANDOM:

        IYA = IYA + 1 ;

        pointer  P = MA(IXA);  adjust pointer to allow for units from
           DB : P = P + [ P/IRA] *IRB ;  store;

        end internal_A .

6)  Loop internal_B:

        follow procedure for internal_A substituting references for
           design  DB  but pointer  P  updated  P = P + [ P/IRB] *IRA+IRA;

        end internal_B ;

7)      end external

        end
```

10.6.5 Randomization control

The randomization procedure in DSIGNX is controlled by an array IY embedded in the main storage area DSIGNDX (§10.5.3). The permutation of IY is controlled by the block structure formula and this algorithm shows how this may be done. The factors are processed in the order of occurrence in the block structure formula. For each factor F the factors in the formula are divided into three disjoint sets: factor F, (external) factors {EF} in which F is nested and the remaining (internal) factors {IF}. For each combination of levels of the external factors the levels of F are permuted randomly or according to the requested (restricted) randomization.

Two units with representation $(k_1, k_2, \ldots, k_p, \ldots k_n)$ and $(k_1, k_2, \ldots, k_p', \ldots k_n)$ are separated in the lexicographical list by a distance $(k_p' - k_p)\ m_p$ (§10.6.2) and a permutation of the levels of factor p changing level k_p to level k_p' causes a

block of units to be 'moved' by this distance. The algorithm tallies, for each cell, the total shift caused by the randomizing process.

Algorithm randomization:

Variables

IY : the permutation array

IRR : restricted randomization indicator

EF,IF : factors 'external' and 'internal' to factor F

IEF,IIF : product of levels for external and internal factors (for controlling loop)

SP : shift in lexicographical location caused by adding 1 to level of factor F

PSHIFT : shift caused by permuting levels of factor F .

1) Initialise permutation array (IY).

2) Loop factor:

select next block factor F; check restriction requested;

if no randomization then end factor; else

check valid and set switch IRR;

form shift pointer $SP = m_p$ (§10.6.2);

3) Use nesting matrix to form sets F, EF, and IF; form IEF, IIF;

Remark: For each combination of the external factors, randomize levels of factor F then calculate pointer shift for all combinations of F and IF.

4) Loop external:

obtain next combination of external factors (routine NEXLEV).

5) Permute levels of factor F as instructed by IRR ;

store permutation in work array IPERM.

6) Loop level_F:

calculate PSHIFT = (IPERM(LEVF) - LEVF) * SP .

7) Loop internal:

obtain next combination of internal factors (routine NEXLEV).

8) Determine current unit (routine NEXLEV) add PSHIFT to associated element in permutation array IY

9) end internal; end level_F; end external;

end factor; end

10.6.6 Restricted randomization

In §7.4 a routine for restricted randomization of a factor with p^m levels was outlined in which

1) a complete array of dimension $m \times p^m$ is selected at random from a set of suitable arrays;

2) the levels of each row are randomly permuted and

3) the array is premultiplied (modulo p) by a random non-singular matrix of order m.

For randomizing factors with 2^3 and 3^2 levels complete arrays are small and those few of interest can be stored easily within the program. For randomizing factors with 16 levels complete arrays can be stored in a random-access file or a small subset held within the program. Steps 1) and 2) above are straightforward. The following algorithm describes the construction of random matrix L^T. The product of L^T and the complete array can be achieved through a minor modification to the routine DSKEY given in Appendix A10.2.

The algorithm works as follows: Array MARKER has p^m cells $(0, 1, \ldots p^m - 1)$ one cell for each possible column; order the columns lexicographically then the ith entry of MARKER denotes whether the column is available for selection (the null column is not). The two-dimensional array STACK is used to build up the group formed by the m column generators and the null vector; it is initialised with the null vector.

Algorithm random_matrix:

Variables

MARKER	: array indicating available columns (length p^m)
STACK(I)	: ith column of array stack, holding element of column subgroup (length p^{m-1})
NS	: number of elements in column subgroup
NR	: number of available columns

LT : the random matrix $(m \times m)$

A,S : the chosen column and its lexicographical location.

1) Initialise:

MARKER ← 0, MARKER(0) = 1, STACK(1) ← 0, NS = 1, NR = $p^m - 1$;

2) Loop column:

choose random integer R in range 1, NR ;

locate Rth available cell in MARKER = S ; convert S to associated column vector A ; store as next column in LT;

3) If last column end else

Remark: extend group by finding direct sums of new column and existing group $(\underline{a} + \underline{c}_o, \underline{a} + c_1 \ldots 2\underline{a} + \underline{c}_o \ldots (p-1)\ \underline{a} + c_o \ldots)$

4) NI = 0 ; N2 = NS ;

5) Loop level (1, 2 ... p - 1):

6) Loop subgroup (1, 2 ... NS):

NI = NI + 1; N2 = N2 + 1;

STACK(N2) = STACK(NI) + A (modulo p);

LOC = lexicographical location of STACK(N2);

MARKER(LOC) = 1; NR = NR - 1 ;

7) end subgroup; end level;

8) NS = N2; end column;

9) end

10.6.7 Output directives

Discussion of output routines is marginal here and we outline only the following routines:

1) checking the layout of the two-dimensional field plan;

2) determining the unit for any cell in the field plan.

The first routine checks, for example, that for the formula (A * B)/C/D blocks defined by factor C fit into blocks defined by stratum A.B . The second routine makes use of this fit to determine the levels of the block factors corresponding to each cell. We use the arrangement that

if $n = rc$ levels of a factor are arranged in an $r \times c$ rectangular grid, then cell (i,j) corresponds to level $ci + j$ of the factor $(0 \leq i \leq r-1,\ 0 \leq j \leq c-1)$.

Algorithm layout:

Variables:

NLF : number of levels of current factor;

NR,NC : number of rows/columns for blocks of current factor;

MR,MC : size of external block;

IR,IC : number of blocks to be fitted into external block.

1) Input dimensions of overall plan (treat as block factor '0');
Check dimensions agree with number of units;

2) Loop in_factor:
input name of next factor;
find position in block structure formula;
read no. of rows NR and columns NC and store;
end in_factor.

Remark: it is convenient to omit specifying a factor with NR = NC = 1 .

3) Check at most one factor unspecified;
if one then set no. rows/columns of factor = 1; else
check minimum row entry = 1, minimum column entry = 1;

Remark: check dimensions of each factor.

4) Loop factor:
for next factor F use nesting matrix (§1.3) to determine set of (external) factors in which it is nested;

5) Determine MR, MC the minimum no. of rows/columns of external factors;

6) Determine MR/NR = IR ; MC/NC = IC ; check IR, IC are integer
check IR * IC = NLF ;

7) end factor; end

Algorithm locate:

Variables:

NBF : number of block factors;

KR,KC : row and column location of cell;

LR,LC : row and column of factor being processed;

FACLEVEL : level of factor being processed (length NBF).

1) Input:

row number KR , column number KC (counting from zero);

2) Loop factor:

(for each block factor in formula) read NR, NC and IC stored in algorithm layout;

3) LR = [KR/NR] ; LC = [KC/NC] ;

KR = KR-LR*NR ; KC = KC-LC*NC ;

FACLEVEL (factor) = LR*IC+LC ; end factor;

4) Calculate lexicographical position of FACLEVEL; end

Appendix 10.1

Common area DSIGNDX.

Common area DSIGNDX outlined in §10.3.1 contains data structures and directories. In this appendix the contents of the cells shown in Fig. 10.1 are detailed. Directories A - F are of fixed size and location as determined by common area POINT . The areas G1 - G8 are data areas and usually disjoint, each new entry being located at the first available location in the storage area.

A. Principal directory: three words by declared object.
   1: location of identifier (G1), zero if absent
   2: type of object: 2 = factor, 4 = block structure, 5 = treatment structure, 6 = design, 7 = heading
   3: location of object in its type directory (B,C,D,E or F).

B. Factor directory: five words per factor.
   1: number of levels
   2: length of name (i.e. as described by directive $NAME)
   3: location of factor name (G7), zero if absent
   4: length of level names
   5: location of level names (G8), zero if absent.

C. Block structure directory: three words per structure.
   1: location of structure formula (G3)
   2: number of factors in structure
   3: dimension of structure.

D. Treatment structure directory: as block directory.

E. Design directory: six words per design (plan)
   1: number of units in unrestricted plan
   2: location of block structure in directory C
   3: location of treatment structure in directory D
   4: location of design (G4)
   5: location of permutation list (G5)
   6: location of unit numbers (G6).

F. Heading directory: two words per heading

1: location of heading (G2)

2: length of heading.

Notes on areas G1 - G8

1) G1, G2, G7, G8 : one character is held per byte.

2) G3 contains an entry for each factor in the block structure giving

   a) the location in the factor directory C and

   b) the row of the nesting matrix in binary form.

3) G4 treatments are held as a number indicating lexicographical location.

4) G5 contains a permutation for deriving the plan from the design.

5) G6 the cells are ordered by units, the unit numbers are arbitrary.

Algorithm for the design key generator.

An algorithm DSKEY for constructing designs by a design key generator (§3.3.1) is presented; it is similar to that contained in DSIGNX but less versatile. (Some options have been omitted to clarify essential features.) The program generates the units in lexicographical order and uses the design key matrix to convert each unit vector into a treatment vector. In the version shown here, as in DSIGNX, the treatment vector is then converted to a number denoting the lexicographical location of the treatment.

The auxiliary algorithm NEXVEC is a combination of DSIGNX subroutines SETLEV and NEXLEV (§10.6.2). It is slightly more general than required by routine DSKEY but the extra options are readily understood and make it suitable for use with table control, randomization etc.

Comment cards in DSKEY indicate changes required if the full treatment vector is to be output.

Structure.

SUBROUTINE DSKEY (KEY, NPP, NPT, LVEC, LV, LCOMB, MPL, IFAULT)

Formal parameters.

KEY	Integer array (NPP, NPT)	Input: design key matrix K
NPP	Integer	Input: number of plot factors
NPT	Integer	Input: number of treatment (pseudo-) factors
LVEC	Integer array (3, LV)	Input: see below Output: " "
LV	Integer	Input: length of LVEC
LCOMB	Integer array (MPL)	Output: treatment combinations stored in compact form
MPL	Integer	Input: maximum length of LCOMB
IFAULT	Integer	Fault indicator

Fault indications.

0: no fault occurred

1: NPP + NPT > LV

2: number of levels or input level for a factor is invalid

3: MPL < number of units

Input and output for LVEC

The array LVEC (§10.6.2) has three rows, denoted here R1, R2 and R3, with elements K_i, L_i and M_i respectively. It is used in routine NEXVEC to control the cycling of plot factors and give the next unit in lexicographical order and in routine DSKEY to convert the treatment vector to a single number for compact storage. The first NPP locations in each row refer to plot factors and the next NPT locations refer to treatment factors.

On input to routine DSKEY plot factor locations of LVEC contain:

R1: current plot factor levels (normally zero) (K_i)

R2: number of levels of plot factors (L_i)

R3: a suitable vector of products (M_i)

Vector R1 is updated to hold the current plot vector. The treatment factor locations of LVEC should contain on input:

R1: treatment base vector $\underset{\sim}{t}_o$ (K_i)

R2: number of levels of treatment (pseudo) factors (L_i)

R3: a suitable vector of products (M_i)

These vectors are not updated.

The auxiliary routine NEXVEC

The normal action of routine NEXVEC is to update values K_i and, if the input value for LOC is non-negative, to calculate a new value for LOC from updated values K_i. On a given call of the routine only one K_i can increase and the parameter NL returns the value of i.

The action of NEXVEC can be modified in several ways:

a) If input value L_j is negative then the jth factor is ignored when updating the R1 vector but not when evaluating LOC . (This feature is used when working with subarrays.)

b) The parameter IND is a switch with the following action:

IND = 0 the values K_i are updated;

IND = 1 all K_i for which $L_i > 0$ are set to zero;

IND = 2 as for IND = 1 but also values M_i are generated from (the absolute values of) levels L_i .

Array LVEC can hold information relating to several multidimensional arrays. The first and last factors for each array are defined by locations NIF and NFF respectively. (The number of factors is NFF - NIF + 1 .)

SUBROUTINE NEXVEC (LVEC, NIF, NFF, LOC, IND, LF, IFAULT)

Formal parameters: (denote the rows of LVEC by R1, R2 and R3)

LVEC	Integer array	Input:	R2 contains the number of levels for each factor
		Output:	R1 contains current levels K_i
NIF	Integer	Input:	location of first factor in LVEC
NFF	Integer	Input:	location of last factor in LVEC
LOC	Integer	Input:	LOC < 0 the location of the cell corresponding to output vector R1 is not calculated; otherwise LOC $\geq$ 0
		Output:	cell location if required
IND	Integer	Input:	IND = 0 R1 is updated; IND = 1 R1 is initialised; IND = 2 R1 and R3 are initialised

LF	Integer	Output: location in LVEC of factor whose level is increased. If IND = 1 or 2 then LF = NFF + 1. If LF = NIF - 1 all adjusted elements of R1 are zero.
IFAULT	Integer	Fault indicator: IFAULT = 0 no fault; IFAULT = 1 invalid input for NIF or NFF ; IFAULT = 2 invalid input for IND .

Input and Output for LVEC and LOC

The input and output for the routine NEXVEC is dependent on the input values of the parameters IND and LOC but may be summarised as follows:

		IND = 0	IND = 1	IND = 2
LOC < 0	Input	R1, R2	R2	R2
	Output	R1	R1	R1, R3
LOC ≥ 0	Input	R1, R2, R3	R2, R3	R2
	Output	R1, LOC	R1, LOC	R1, R3, LOC

```
      SUBROUTINE DSKEY(KEY,NPP,NPT,LVEC,LV,LCOMB,MPL,IFAULT)
C
C   A ROUTINE FOR CONSTRUCTING FACTORIAL DESIGNS BY THE
C   DESIGN KEY CONSTRUCTION OF PATTERSON(1976).
C
      DIMENSION KEY(NPP,NPT),LVEC(3,LV),LCOMB(MPL)
C
C    ALTERNATIVE...
C       DIMENSION KEY(NPP,NPT),LVEC(3,LV),LCOMB(NPT,MPL)
C
      IFAULT=1
      NIF=NPP+1
      NFF=NPP+NPT
      IF(NFF.GT.LV)RETURN
      IFAULT=2
      DO 10 I=1,NFF
      IF(LVEC(2,I).LE.0)RETURN
      IF(LVEC(1,I).LT.0.OR.LVEC(1,I).GE.LVEC(2,I))RETURN
10    CONTINUE
      NPL=1
      DO 20 I=1,NPP
20    NPL=NPL*LVEC(2,I)
      IFAULT=3
      IF(NPL.GT.MPL)RETURN
      IFAULT=0
C
      IND=0
      LOC=-1
      DO 60 N=1,NPL
      NOT=0
      DO 50 I=1,NPT
      IT=NPP+I
      K=LVEC(1,IT)
      DO 40 J=1,NPP
40    K=K+KEY(J,I)*LVEC(1,J)
      KA=MOD(K,LVEC(2,IT))
      NOT=NOT+KA*LVEC(3,IT)
50    CONTINUE
      LCOMB(N)=NOT
C
C...OR REPLACE PREVIOUS THREE CARDS BY..
C       LCOMB(I,N)=KA
C 50    CONTINUE
C
      CALL NEXVEC(LVEC,1,NPP,LOC,IND,LF,IFAULT)
60    CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE NEXVEC(LVEC,NIF,NFF,LOC,IND,LF,IFAULT)
C
C  THIS ROUTINE IS USEFUL WHEN SIMULATING MULTI-DIMENSIONAL
C  ARRAYS IN ONE DIMENSION. THE STANDARD OPERATION OF THE
C  ROUTINE IS TO CAUSE THE FIRST ROW OF LVEC TO BE UPDATED.
C  IT HAS SEVERAL MODES OF OPERATION DETERMINED BY THE
C  VALUES OF THE PARAMETERS IND AND LOC.
C
      DIMENSION LVEC(3,NFF)
      IFAULT=1
      IF(NIF.LE.0.OR.NIF.GT.NFF)RETURN
      IFAULT=2
      IF(IND.LT.0.OR.IND.GT.2)RETURN
      IFAULT=0
      IF(IND-1)25,15,5
C
C INITIALISATION ROUTINE
C
5     ITEM=1
      I=NFF+1
      DO 10 K=NIF,NFF
      I=I-1
      LVEC(3,I)=ITEM
10    ITEM=ITEM*IABS(LVEC(2,I))
C ...AND R1
15    DO 20 K=NIF,NFF
      IF(LVEC(2,K).GT.0)LVEC(1,K)=0
20    CONTINUE
      LF=NFF+1
      GO TO 50
C
C FIND NEXT CELL
C
25    LF=NFF
30    IF(LVEC(2,LF).LT.0)GO TO 40
      LVEC(1,LF)=LVEC(1,LF)+1
      IF(LVEC(1,LF).LT.LVEC(2,LF))GO TO 50
      LVEC(1,LF)=0
40    LF=LF-1
      IF(LF.GE.NIF)GO TO 30
C
50    IF(LOC.LT.0)GO TO 70
C
C DETERMINE LOCATION
      LOC=0
      DO 60 I=NIF,NFF
60    LOC=LOC+LVEC(1,I)*LVEC(3,I)
70    RETURN
      END
```

REFERENCES

Addelman,S.(1962a) Symmetrical and asymmetrical fractional factorial plans. Technometrics,4,47-58.

Addelman,S.(1962b) Orthogonal main-effect plans for asymmetrical factorial experiments. Technometrics,4,21-46.

Addelman,S.(1965) The construction of a 2**(17-9) resolution V plan in eight blocks of 32. Technometrics,7,439-443.

Addelman,S. and Kempthorne,O.(1961) Some main effect plans and orthogonal arrays of strength 2. Ann. Math. Statist., 32,1167-1176.

Anderson,D.A. and Thomas,A.M.(1979) Near minimal resolution IV designs for the s**n factorial experiment. Technometrics, 21,331-336.

Alvey,N.G et al(1977) GENSTAT,a general statistical program. Rothamsted Experimental Station,Harpenden,Hertfordshire,U.K.

Aschbacher,M.(1971) On collineation groups of symmetric block designs, J. Combinat. Theory,A,11,272-281.

Ash,A. and Hedayat,A.(1978) An introduction to design optimality with an overview of the literature. Commun. Statist.- Theor. Meth., A7(14),1295-1325.

Atkinson,A.C.(1969) The use of residuals as a concomitant variable. Biometrika,1969,56,33-41.

Bailey,R.A.(1978) Patterns of confounding in factorial designs. Biometrika,64,597-603

Bailey, R.A., Gilchrist,F.H.L., and Patterson,H.D.(1977) Identification of effects and confounding patterns in factorial designs. Biometrika,64,347-354.

Baker,R.J. and Nelder,J.A.(1978) The GLIM system,Release 3, Numerical Algorithms Group,Oxford.

Bartlett,M.S.(1938) The approximate recovery of information from field experiments with large blocks. J. Agric. Sci.,28, 418-427.

Bartlett,M.S.(1978) Nearest neighbour models in the analysis of field experiments. J. R. Statist. Soc.,B,40,147-174.

Berenblut,I.I.(1970) Treatment sequences balanced for the linear component of residual effects. Biometrics,26,154-156.

Binet,F.E.,Leslie,R.T.,Wiener,S. and Anderson,R.L.(1955). Analysis of confounded factorial experiments in single replication. North Carolina Agric. Exp. Sta. Tech. Bull. 113.

Bose,R.C.(1939) On the construction of balanced incomplete block designs. Ann. Eugen.,9,353-399.

Bose,R.C.(1947) Mathematical theory of the symmetrical factorial design, Sankhya,8,107-166.

Bose,R.C. and Bush,K.A.(1952) Orthogonal arrays of strength two and three. Ann. Math. Statist.,23,508-524

Bose,R.C.,Clatworthy,W.H. and Shrikande,S.S.(1954) Tables of partially balanced designs with two associate classes. North Carolina Agric. Exp. Sta. Tech. Bull. 107.

Bose,R.C. and Kishen,K.(1940) On the problem of confounding in the general symmetric factorial design. Sankhya,5,21-36.

Bose,R.C. and Nair,K.R.(1939) Partially balanced incomplete block designs. Sankhya,4,337-372.

Bose,R.C. and Shimamoto,T.(1952) Classification and analysis of designs with two associate classes. J. Amer. Statist. Ass. ,47,151-184.

Bose,R.C.,Shrikande,S.S. and Bhattacharya,K.N.(1953) On the construction of group divisible incomplete block designs. Ann. Math. Statist.,24,167-195.

Bose,R.C.,Shrikande,S.S. and Parker,E.T.(1960) Some further results on the construction of mutually orthogonal Latin squares and the falsity of of Euler's conjecture. Can. J. Math.,12,189-203.

Box,G.E.P. and Behnken,P.W.(1960) Simplex sum designs:A class of second order rotatable designs derivable from those of first order. Ann. Math. Statist.,31,838-864.

Box,G.E.P. and Hunter,J.S.(1961) The 2**k-p fractional factorial designs. Technometrics ,3:I 311-352:II 449-458.

Box,G.E.P.,Hunter,W.G. and Hunter,J.S.(1978) Statistics for experimenters. New York:John Wiley and Sons Inc.

Box,G.E.P. and Jenkins,G.M.(1976) Times series analysis: forecasting and control. Rev. ed. San Francisco:Holden-Day.

Box,G.E.P. and Wilson,K.J.(1951) On the experimental attainment of optimum conditions. J. R. Statist. Soc.,B,13,1-45.

Brenna,L.S. and Kramer,C.Y.(1961) Factorial treatments in rectangular lattice designs. J. Amer. Statist. Ass.,56,368-378.

Causey,B.D.(1968) Some examples of multidimensional incomplete block designs. Ann. Math. Statist.,39,1577-1590.

Chambers,J.M.(1971) Regression updating. J. Amer. Statist. Ass.,66,744-748.

Chang,L. and Liu,W.(1964) Incomplete block designs with square parameters for which k<10 and r<10. Scientia Sinica, 13,1493-1495.

Chang,L.,Liu,C. and Liu,W.(1965) Incomplete block designs with triangular parameters for which k<10 and r<10. Scientia Sinica,14,329-338.

Claringbold,P.J.(1969) An approach to conversational statistics. In Statistical Computation(R.C.Milton and J.A.Nelder,eds) pp 267-283, New York:Academic Press.

Clatworthy,W.H.(1955) Partially balanced incomplete block designs with two two associate classes and two treatments per block. J. Res. Nat. Bur. Stand.(U.S.),54,177-190.

Clatworthy,W.H.(1956) Contributions on partially balanced incomplete block designs with two associate classes. U.S. Nat. Bur. Stand. Appl. Math. Ser.,47

Clatworthy,W.H.(1967) Some new families of partially balanced designs of the Latin square type and related designs. Technometrics,9,229-244

Clatworthy,W.H(1973) Tables of two-associate class partially balanced incomplete block designs. U.S. Nat. Bur. Stand. Appl. Math. Ser.,63

Cochran,W.G. and Cox,G.M.(1957) Experimental Designs,2nd ed., New York:John Wiley and Sons Inc.

Cochran,W.G.,Autrey,K.M. and Cannon,C.Y.(1941) A double change-over design for dairy cattle feeding experiments. J. Dairy Sci.,24,937-951.

Conniffe,D. and Stone,J.(1974) The efficiency factor of a class of incomplete block designs Biometrika,61,633-6.

Connor,W.S. and Young,S.(1961) Fractional factorial designs for experiments with factors at two and three levels. U.S. Nat. Bur. Stand. Appl. Math. Ser.,58.

Cook,R.D. and Nachtsheim,C.J.(1980) A comparison of algorithms for constructing exact D-optimal designs. Technometrics,22,315-323.

Cotter,S.C.(1978) Block designs for fractional factorial experiments with replication, J. R. Statist. Soc.,B,40,1-78.

Cotter,S.C.,John,J.A. and Smith,T.M.F.(1973) Multi-factor experiments in non-orthogonal designs. J. R. Statist. Soc.,B, 361-367.

Cox,D.R.(1956) A note on weighted randomization. Ann. Math. Statist.,27,1144-1150.

Cox,D.R.(1958) Planning of experiments, New York:John Wiley and Sons.

Daniel,C.(1976) Applications of statistics to industrial experimentation. New York:John Wiley and Sons.

Das,M.N.(1964) A somewhat alternative approach for construction of symmetrical factorial designs and obtaining maximum number of factors. Calcutta Statist. Ass. Bull.,13,1-17.

Das, M.N. and Narasimham, V.L. (1962) Construction of rotatable designs through BIB designs., Ann. Math. Stat., 33, 1421-1439

David,H.A.(1963) The structure of cyclic paired-comparison designs. J. Aust. Math. Soc.,3,117-127.

David,H.A. and Wolock,F.W.(1965) Cyclic designs. Ann. Math. Statist.,36,1526-1534

Davies,O. ed.(1978) Design and analysis of industrial experiments. London:Longman.

Davis,A.W. and Hall,W.B.(1969) Cyclic change-over designs. Biometrika,56,283-293.

Dean.A.M. and John,J.A.(1975) Single replicate factorial experiments in generalised cyclic designs:II asymmetrical arrangements. J. R. Statist. Soc.,B,63-76.

Dean,A.M. and Lewis,S.M.(1980) A unified theory for GC/n designs. J. Statist. Plan. and Infer.,4,13-24

Debaun,R.(1959) Response surface designs for three factors at three levels. Technometrics,1,1-8.

Denes,J. and Keedwell,A.D.(1974) Latin squares and their applications. London:English Universities Press.

Dyke,G.V.(1964) Restricted randomization for blocks of 16 plots. J. Agric. Sci.,62,215-217.

Dyke,G.V. and Shelley,C.F.(1976) Serial designs balanced for effects of neighbours on both sides. J. Agric. Sci.,87,303-305.

Eccleston,J.A. and Jones,B.(1980) Exchange and interchange procedures to search for optimal row-and-column designs. J. R. Statist. Soc.,B,42,238-243

Fedorov,V.V.(1972) Theory of optimal experiments. Translated by W.J.Studden and E.M.Klimko. New York:Academic Press.

Finney,D.J.(1945) The fractional replication of factorial arrangements. Ann. Eugen.,12,291-301.

Finney,D.J.(1956) Cross-over designs in bioassay. Proc. R. Soc. ,B,145,42-61.

Finney,D.J.(1960) The theory of experimental design. Chicago :Univ. of Chicago Press.

Finney,D.J. and Outhwaite,A.D.(1956) Serially balanced sequences in bioassay. Proc. R. Soc.,B,145,493-507.

Fisher,R.A.(1935) The design of experiments. Edinburgh:Oliver and Boyd

Fisher,R.A. and Yates,F.(1963) Statistical tables for biological agricultural and medical research. 6th ed. Edinburgh:Oliver and Boyd.

Fowlkes,E.B. and Lee,E.T.(1971) Appendix C to Roy,S.N.,Gnanadesikan, R. and Srivastava,J.N. Analysis and design of certain quantitative multiresponse experiments. Oxford: Pergamon Press.

Franklin,M.F. and Bailey,R.A.(1977) Selection of defining contrasts and confounded effects in two-level experiments. Appl. Statist., 26,321-326.

Franklin,M.F. and Mann,A.D.(1980) DSIGNX,Inter-university/ Research Councils Ser.,48.

Franklin,M.F. and Patterson,H.D.(1978) A computer program for the construction of experimental plans. Proceed. COMPSTAT conference, Leiden.

Freeman,G.H.(1957) Some further methods of constructing regular group divisible incomplete block designs. Ann. Math. Statist.,28,479-487.

Freeman,G.H.(1958) Families of designs for two successive experiments. Ann. Math. Statist.,29,1063-1078.

Freeman,G.H.(1976a) A cyclic method of constructing regular group divisible incomplete block designs.Biometrika,63,555-8.

Freeman,G.H.(1976b) On the selection of designs for comparative experiments. Biometrics,32,195-199.

Freeman,G.H.(1979) Complete Latin squares and related experimental designs. J. Roy. Statist. Soc.,B,253-262

Fry,R.E.(1961) Finding new fractions of factorial experimental designs. Technometrics,3,359-370.

Galil,Z. and Kiefer,J.(1980) Time and space-saving computer methods, related to Mitchell's DETMAX, for finding D-optimum designs. Technometrics,22,301-313.

Gentleman,W.M.(1974) Basic procedures for large sparse or weighted linear least squares problems. Algorithm AS75, Appl. Statist.,23.

Gower,J.C.(1968) Simulating multi-dimensional arrays in one dimension. Algorithm AS1,Appl. Statist.,17,180-185

Gower,J.C.(1969) Remark ASR1 on AS1 subroutine package. Appl. Statist.,18,116

Greenfield,A.A.(1976) Selection of defining contrasts in two -level experiments. Appl. Statist.,25,64-67.

Grundy,P.M. and Healy,M.J.R.(1950) Restricted randomization and quasi-Latin squares. J. R. Statist. Soc.,B,12,286-291.

Hall,M.Jr.,Lane,R. and Wales,D.(1970) Designs derived from permutation groups. J. Combinat. Th.,8,12-22.

Hall,W.B. and Williams,E.R.(1973) Cyclic superimposed designs. Biometrika,60,47-53.

Harshbarger,B.(1949) Triple rectangular lattices. Biometrics, 5,1-13.

Harville,D.A.(1975a) Experimental randomization:who needs it ! Amer. Statist.,29,27-31.

Harville,D.A.(1975b) Computing optimum designs for covariance models. pp209-228 in Statistical design and linear models ed. Srivastava,J.N., Amsterdam:North Holland Publishing Co.

Hedayat,A. and Asfarinejad,K.(1975) Repeated measurements designs:I. in Statistical design and linear models. Ed. Srivastva,J.N. Amsterdam:North Holland Publishing Co.

Hedayat,A. and Asfarinejad,K.(1978) Repeated measurements designs:II Ann. Statist.,6,619-628.

Hedayat,A. and Seiden,E.(1974) On the theory and application of sum composition of Latin squares and orthogonal Latin squares. Pacific J. Math.,54,85-113

Hedayat,A,Seiden,E and Federer,W.T.(1972) Some families of designs for multistage experiments:mutually balanced Youden designs when the number of treatments is prime power or twin primes I. Ann. Math. Statist.,43, 1517-1527

Hedayat,A. and Shrikande,S.S.(1971) Experimental designs and combinatorial systems associated with Latin squares and sets of mutually orthogonal Latin squares. Sankhya,A,423-432.

Hedayat,A. and Wallis,W.D.(1978) Hadamard matrices and their applications. Ann. Statist.,6,1184-1238.

Herzberg,A.M. and Cox,D.R.(1969) Recent work on the design of experiments. J. R. Statist. Soc.,A,132,29-67.

Hill,W.J. and Hunter,W.G.(1966) A review of response surface methodology:a literature survey. Technometrics,8,571-590.

Hoblyn,T.N.,Pearce,S.C. and Freeman,G.H.(1954) Some considerations in the design of successive experiments in fruit plantations. Biometrics,10,503-515

Hoke,A.T.(1974) Economical second-order designs based on irregular fractions of the 3**n factorial. Technometrics,16, 375-384.

Jarrett,R,G. and Hall,W.B.(1978) Generalized cyclic incomplete block designs. Biometrika,65,397-401

Jarrett,R.G.(1977) Bounds for the efficiency factors of block designs. Biometrika,64,67-72.

Jenkyn,J.F.,Bainbridge,A,Dyke,G.V. and Todd,A.D.(1979) An investigation into inter-plot interactions,in experiments on barley,using balanced designs. Ann. appl. Biol., 92,11-28.

John,J.A.(1966) Cyclic Incomplete block designs. J. R. Statist. Soc., B,28,345-360

John,J.A.(1973) Generalised cyclic designs in factorial experiments. Biometrika,60,55-63.

John,J.A.(1979) On the efficiency factors of generalised cyclic designs in factorial experiments.Submitted to Ann. Statist.

John,J.A.(1980) Factorial-balance and the analysis of designs with factorial structure. Submitted for publication.

John,J.A.(1981) Efficient cyclic designs J. R. Statist. Soc., B,43,76-80

John,J.A. and Mitchell,T.J.(1977) Optimal incomplete block designs. J. R. Statist. Soc.,B,39-43

John,J.A. and Dean,A,M.(1975) Single replicate factorial experiments:I symmetrical arrangements J. R. Statist. Soc., B,37,63-76.

John,J.A. and Smith,T.M.F.(1972) Two-factor experiments in non-orthogonal designs. J. R. Statist. Soc.,B,401-409

John,J.A. and Turner,G.(1977) Some new group devisible designs. J. Statist. Plan. and Inf.,1,103-107

John,J.A.,Wolock,F.W. and David,H.A.(1972) Cyclic Designs U.S. Nat. Bur. Stand. Appl. Math. Ser.,62

John,P.W.M.(1971) Statistical design and analysis of experiments. New York:The Macmillan Company.

Jones,B.(1976) An algorithm for deriving optimal block designs. Technometrics,18,451-458.

Kageyama,S.(1972) A survey of resolvable solutions of balanced incomplete block designs. Int. Statist. Rev.,40,269-273

Kempthorne,O.(1953) A class of experimental designs using blocks of two plots. Ann. Math. Statist.,24,946-967

Kempthorne,O.(1952) The design and analysis of experiments. New York:John Wiley and Sons.

Kempthorne,O.(1977) Why randomise! J. Statist. Plan. and Inf.,1,1-17

Kennard,R.W. and Stone,L.A.(1969) Computer-aided design of experiments. Technometrics,11,137-148.

Kiefer,J.(1959) Optimum experimental designs. J. R. Statist. Soc.,B,21,272-319

Kiefer,J.(1975) Construction and optimality of generalized Youden designs. Statistical design and Linear Models. Ed. Srivatava,J.N. Amsterdam:North Holland Pub. Co.

Kiefer,J. and Wolfowitz,J.(1959) Optimum designs in regression problems. Ann. Math. Statist.,30,271-294

Kishen,K. and Srivastava,J.N.(1959) Mathematical theory of confounding in asymmetrical and symmetrical factorial designs. J. Indian Soc. Agric. Statist.,11,73-110

Knuth,D.E.(1969) The art of computer programming:Vol 2 seminumerical algorithms. Massachusetts:Addison Wesley.

Kramer,C.Y. and Bradley,R.A.(1957) Examples of intra-block analysis for factorials in group divisible partially balanced incomplete block designs. Biometrics,13,197-224.

Kurkjian,B. and Zelen,M.(1963) Applications of the calculus of factorial arrangements:I block and direct product designs. Biometrika,50,63-73

Lewis,S.M. and John,J.A.(1976) Testing main effects in fractions of asymmetrical factorial experiments. Biometrika, 63,678-80

Li,J.C.R.(1944) Design and statistical analysis of some confounded factorial experiments. Iowa Agr.Exp.Stat.Res.Bul.333

Margolin,B.H.(1969a) Resolution IV fractional factorial designs. J. R. Statist. Soc.,B,514-523

Margolin,B,H.(1969b) Orthogonal main-effect plans permitting estimation of all two-factor interactions for the 2**n3**m factorial series of designs. Technometrics,11,747-762

Masuyama,M.(1965) Cyclic generation of triangular PBIB designs. Rep. Statist. Appl. Res.,JUSE,12,73-81

Mitchell,T.J.(1974) An algorithm for the construction of D-optimal experimental designs. Technometrics,16,203-210

National Bureau of Standards.(1957) Fractional factorial experiment designs for factors at two levels. Appl. Math. Ser.,48

National Bureau of Standards.(1959) Fractional factorial experiment designs for factors at three levels. Appl. Math. Ser.,54.

Nelder J.A.(1965) The analysis of randomised experiments with orthogonal block structure. I Block structure and the null analysis of variance. II Treatment structure and the general analysis of variance. Proc. R.Soc.,A,283,163-178

Patterson,H.D.(1951) Change-over trials J. R. Statist. Soc., B,13,256-271

Patterson,H.D.(1965) The factorial combination of treatments in rotation experiments. J. Agric. Sci.,65,171-182

Patterson,H.D.(1973) Quenouille's changeover designs. Biometrika,60,33-45

Patterson,H.D.(1976) Generation of factorial designs. J. R. Statist. Soc.,B,175-179

Patterson,H.D. and Bailey R.A.(1978) Design keys for factorial experiments. Appl. Statist,27,335-343

Patterson,H.D. and Lucas,H.L.(1962) Change-over designs. North Carolina Agric. Exp. Sta. Tech. Bull. 147

Patterson,H.D. and Silvey,V.(1980) Statutory and recommended list trials of crop varieties in the United Kingdom. J. R. Statist. Soc.,A,219-252

Patterson,H.D. and Williams E.R.(1975) Some theoretrical results on general block designs. Proc. 5th Brit. Combinat. Conf. 489-496

Patterson,H.D. and Williams E.R.(1976) A new class of resolvable incomplete block designs. Biometrika,63,83-92

Patterson,H.D.,Williams,E.R. and Hunter E.A.(1978) Block designs for variety trials. J. Agric. Sci.,90,395-400

Pearce S.C(1963) The use and classification of non-orthogonal designs(with discussion). J. R. Statist. Soc.,A,126,353-377

Pearce,S.C.(1968) The mean efficiency of equi-replicate designs, Biometrika,55,251-3

Pearce,S.C.(1970) The efficiency of block designs in general. Biometrika,57,339-346

Pearce,S.C.,Calinski,T. and Marshall,T.F.de C.(1974) The basic contrasts of an experimental design with special reference to the analysis of data. Biometrika,61,449-460

Pesotan,H.,Raktoe,B.L. and Worthley,R.A.(1978) On main effect fold-over designs.J. Statist Plan. and Inf.,2,277-292

Plackett,R.L.(1946) Some generalizations in the multifactorial design. Biometrika,33,328-333

Plackett,R.L.(1966) Random permutations. J. R. Statist. Soc., B,30,517-534

Plackett,R.L. and Burman,J.P.(1946) The design of optimum multifactorial experiments. Biometrika,33,305-325

Pocock,S.J. and Simon,R.(1975) Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial. Biometrics,31,103-115

Pothoff.R.F.(1963) Some illustrations of four dimensional incomplete block constructions. Calcutta Statist. Ass. Bull., 12,19-30

Preece,D.A.(1966) Some row and column designs for two sets of treatments. Biometrics,22,1-25

Preece,D.A.(1971) Some new balanced row-and-column designs for two non-interacting sets of treatments. Biometrics,27, 426-430

Preece,D.A.(1972) Combinatorial analysis and experimental design. The Statistician,21,77-87

Raghavarao,D.(1971) Constructions and combinatorial problems in design of experiments. New York:John Wiley and Sons.

Raktoe,B.L. and Federer,W.T.(1972) Construction of confounded mixed factorial and mixed lattice designs. Aust. J. Statist.,14,25-36

Rao,C.R.(1961a) A study of BIB designs with replications 11 to 15. Sankhya,23,117-127

Rao,C.R.(1961b) Generation of random permutations of given number of elements using random sampling numbers. Sankhya,23,305-307

Rogers,C.E.(1973) Interpreting structure formulae. Algorithm AS65,Appl. Statist.,22,414-424

St.John,R.C. and Draper,N.R.(1975) D-optimality for regression designs:a review. Technometrics,17,15-23

Sampford,M.R.(1957) Methods of construction and analysis of serially balanced sequences. J. R. Statist. Soc.,21,286-304

Shah,B.V.(1958) On balancing in factorial experiments. Ann. Math. Statist.,29,766-779

Shrikande,S.S.(1965) On a class of partially balanced incomplete block designs. Ann. Math. Statist.,36,1807-1814.

Shrikande,S.S. and Raghavarao,D.(1963) A method of construction of incomplete block designs. Sankhya,A,25,399-402

Srivastava,J.N.(1978) A review of some recent work on discrete optimal factorial designs for statisticians and experimenters. pp267-329 in Developments in statistics,Vol 1, New York:Academic Press.

Starks,T.H.(1964) A note on small orthogonal main-effect plans for factorial experiments. Technometrics,6,220-222.

Sutter,G.J. Zyskind,G. and Kempthorne,O.(1963) Some aspects of constrained randomization. Aeronaut.Res.Lab.Rep.ARL 63-18.

Takeuchi,K.(1962) A table of difference sets generating balanced incomplete block designs. Rev. Inst. Internat. Statist., 30,361-366

Tolmie,J.(1973) User's manual for the design construction package DSIGN. Issued by ARC Unit of Statistics,Edinburgh.

Vajda,S.(1967) The mathematics of experimental design: incomplete block designs and Latin squares. London:Griffin.

White,D. and Hulquist,R.A.(1965) Construction of confounding patterns for mixed factorial designs. Ann. Math. Statist.,36,1256-71

Wilkinson,G.N.(1970) A general recursive procedure for analysis of variance. Biometrika,57,19-46

Wilkinson,G.N. and Rogers,C.E.(1973) Symbolic description of factorial models for analysis of variance. Appl. Statist.,22, 392-399

Williams,E.J.(1949) Experimental designs balanced for the estimation of residual effects of treatments. Aust. J. Sci. Res. A,2,149-168

Williams,E.R.(1975) A new class of resolvable block designs. Ph.D Thesis,University of Edinburgh.

Williams,E.R. and Patterson H.D.(1977) Upper bounds for efficiency factors in block designs. Aust. J. Stat.,19,194-201.

Williams,E.R.,Patterson,H.D. and John,J.A.(1976) Resolvable designs with two replications. J. R. Statist. Soc.,B,38,296-301.

Williams,E.R.,Patterson H.D. and John,J.A.(1977) Efficient two-replicate resolvable designs. Biometrics,33,713-717

Williams,R.M.(1952) Experimental designs for correlated observations. Biometrika,39,151-167

Wynn,H.P.(1972) Results in the theory and construction of D-optimum experimental designs.J. R. Statist. Soc.,B,34,133-147

Yates,F.(1933) The principles of orthogonality and confounding in replicated experiments J. Agric. Sci.,23,108-145.

Yates,F.(1936a) Incomplete randomised blocks. Ann. Eugen.,7, 121-140

Yates,F.(1936b) A new method of arranging variety trials involving a large number of varieties.J. Agric. Sci.,26,424-455

Yates,F.(1937) The design and analysis of factorial experiments. Imperial Bureau of Soil Science,Harpenden.

Yates,F.(1937) A further note on the arrangement of variety trials:quasi-Latin squares. Ann. Eugen. 7,319-332

Yates,F.(1970) Experimental design:selected papers. London:Charles Griffin and Co. (Also contains Yates(1933,1936a,b))