α_n-Designs

A thesis presented to

The University of Waikato

in fulfilment of the requirement for the degree

of

Doctor of Philosophy

by

Katya Ruggiero

The
University
of Waikato
Te Whare Wānanga
o Waikato

2000

Abstract

This thesis defines a broad class of resolvable incomplete block designs for multi-factor experiments, called α_n-*designs*. A general methodology for their construction is described and is shown to be a natural extension of the method used by Patterson and Williams (1976) in their construction of α-designs. In fact, the class of α-designs are a special case of α_n-designs with $n = 1$.

The family of α_n-designs was primarily developed for their use in factorial experiments. While they are particularly suitable for this purpose, for some combinations of design parameters they also provide more efficient designs than the best available α-designs.

Algorithms for the generation of efficient designs require computationally expensive eigenvalue calculations. Therefore, considerable attention is devoted to the study of the structural properties of α_n-designs and how these can be used to develop computationally fast algorithms. In this endeavour it is shown that the treatment concurrence and information matrices have an important property known as B^nC-structure. This is used to derive a closed form mathematical expression for the average efficiency factor of any generalized interaction. It is further shown that this expression can be written recursively. These results, combined with the fact than an α_n-design is completely specified by a small generating array, mean that computer search algorithms can be developed to quickly find highly efficient designs.

A desirable feature in a factorial design is the property of orthogonal factorial structure. The family of α_n-designs do not, in general, admit this property. However, another class of n-dimensional cyclic designs, known as *n-cyclic* designs, suitable for non-resolvable designs for factorial experiments are also considered in this thesis. They possess orthogonal factorial structure and include a sizeable subset of designs which are resolvable. These resolvable n-cyclic designs are shown to be α_n-designs.

The discovery and subsequent development of the family of α_n-designs has stimulated a new set of interesting questions in relation to factorial experiments. These are discussed at the end of the thesis.

Acknowledgements

Each journey involves a number of cross-roads at which we must decide which road to take, and with hindsight we might choose some roads differently. However, in this journey there are two choices that I would not change: my area of research and my supervisor, Professor J.A. (Nye) John. I would like to thank him for his support and encouragement, and his many constructive comments which greatly enhanced the readability of this thesis.

I would like to thank Dr Alec Zwart for his continuous encouragement. His LaTeX support was invaluable.

I also wish to acknowledge and thank my parents who supported and provided me with the opportunities that have enabled me to pursue my studies.

Finally, I would especially like to thank my husband and best friend, Glenn, for having enough belief in me to leave behind the security of our lives in Australia so that I could pursue this PhD program. His support and encouragement helped me to complete this journey.

This work was supported by a grant from the Marsden Fund.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Factorial experiments

> *Nature ... will best respond to a logical and carefully thought out questionnaire; indeed, if we ask her a single question, she will often refuse to answer until some other topic has been discussed.*
>
> R.A. Fisher (1926)

The above quote, extracted from his 1926 paper *The arrangement of field experiments*, was Fisher's response to a widely held view at that time that experiments should be carried out "varying the levels of the essential conditions *only one at a time*". He believed that in many experimental situations a number of factors will impact on any given quality characteristic of the system under investigation and that these factors "must always be considered as potentially interacting with one another" (Fisher, 1935). Hence, the causal relationship between these factors and the system characteristic under investigation should be studied simultaneously in a factorial experiment.

Fisher understood that factorial experiments are an efficient means of acquiring knowledge, where "by more efficient we mean that more knowledge and a higher degree of precision are obtainable by the same number of observations". Factorial experiments are now used across a wide range of disciplines.

Williams and Matheson (1994) consider the design of a factorial experiment to study the effects of four pre-treatments on the germination of six *Acacia mangium*

seedlots. The pre-treatments were applied to seeds which were laid out on petri dishes. The dishes were then placed in a three-tiered germination cabinet. Each tier, or shelf, accommodated twenty-four dishes arranged in six rows by four columns.

The design used by Williams and Matheson is shown in Table 1.1. Each seedlot by pre-treatment factorial treatment combination is represented by a 2-tuple, where the first digit corresponds to seedlot and the second to pre-treatment. Here, the labels $0, 1, \ldots, 5$ have been used to represent the six seedlots and $0, 1, 2$ and 3 for the four pre-treatments. The columns of the design represent blocks. This experiment will not only allow the experimenter to answer questions about differences between seedlots and between pre-germination treatments, but also whether pre-germination treatment affects different seedlots in different ways.

Table 1.1: Factorial design used by Williams and Matheson (1994)

Top shelf				Middle shelf				Bottom shelf			
31	12	50	43	42	00	03	31	23	41	22	50
40	33	11	52	12	33	21	50	40	01	03	42
03	20	41	02	23	41	40	02	31	32	43	10
22	51	00	13	30	52	51	13	52	53	30	21
10	42	23	21	53	11	32	20	00	13	51	02
53	01	32	30	01	22	10	43	12	20	11	33

It can be seen from the design in Table 1.1 that there are numerous ways in which the treatments could have been allocated to blocks. The larger the size of the experiment, the greater the number of possible enumerations; where size is defined by the number of treatments, v, the number of experimental units per block or block size, k, and the number of replications of each treatment, r. Since within block treatment comparisons are more accurately made than comparisons among blocks, the aim is to find a configuration that allows the treatment effects to be precisely estimated. The design in Table 1.1, for example, will allow both main effects to be estimated with optimum precision since all six seedlots are represented in every block and each pre-treatment occurs either once or twice per block.

When the allocation of treatments to blocks is such that the blocks can then be grouped together so that every treatment combination is represented exactly once within a group, the design is said to be *resolvable*. The design in Table 1.1

is resolvable since all twenty-four factorial treatment combinations are represented in the group of four blocks on each shelf. In this case the resolvable structure was used to control for possible variation in conditions among the shelves of the germination cabinet. However, resolvable designs are also useful in practice when the entire experiment cannot be completed at one time. Then, the experiment can be conducted in stages. If for any reason the experiment must be terminated prematurely, then all treatment combinations will have occurred equally often.

Williams and John (1996) give a general algorithm for the construction of resolvable incomplete block designs for factorial experiments; see section 1.5. Beyond this, however, the literature is quite sparse in terms of general methodologies for the construction of this class of designs. While computer algorithms can be used, construction of efficient medium to large designs, such as those that are used in plant-breeding trials for example, require algorithms that are computationally expensive. Hence, the availability of a family of designs with a well-defined method of construction and structural properties that are theoretically well-understood would be highly desirable. The existence of such a family would substantially reduce the design search space so that efficient designs could quickly be found.

The purpose of this thesis is to introduce a family of resolvable incomplete block designs for factorial experiments, and to study their properties. We have called these designs α_n-*designs* since their method of construction and many of their structural properties are an extension of the α-designs of Patterson and Williams (1976). They are a flexible family of designs which exist for a broad range of the design parameters v, k and r. Furthermore, they share an important structural feature with their parent α-designs which can be exploited to considerably reduce the computational effort required to find efficient designs.

1.2 Resolvable designs

One of the early applications of resolvable designs was in plant breeding trials. Plant breeders wanted to be able to test a large number of genetic lines and then make comparisons between all pairs of lines with equal precision. Complete block designs

were considered inappropriate since they require large block sizes. This resulted in experiments that were difficult to manage and large experimental error variance due to non-homogeneous experimental units. Resolvable incomplete block designs were a more attractive alternative since the smaller block size increased the precision with which comparisons could be made while maintaining equal, or near equal, precision of all pairwise comparisons.

A number of families of resolvable incomplete block designs are documented in the literature. Yates (1936), for instance, introduced square lattice designs and a decade later Harshbarger (1947) introduced the rectangular lattice designs. (Lattice designs are known to be optimal because their method of construction ensures that the within block treatment concurrences are as equal as possible.) Lattice designs exist for only a limited number of combinations of the design parameters. In 1967, David defined a class of resolvable cyclic designs. Finally, in 1976 Patterson and Williams introduced the family of α-designs which greatly enhanced the space of available resolvable incomplete block designs.

Lattice and α-designs are well-suited to single factor experiments. Their properties are theoretically well-understood, and they are simple to analyse. For factorial experiments however, as Lewis and Dean (1980) point out, "*one of the main difficulties in using any of these designs is that the assignment of labels to factorial treatment combinations is unclear in terms of a general method for obtaining efficient designs*". They identified a subset of resolvable n-cyclic designs and showed that these designs could be generated for any r that is a multiple of k. John and Ruggiero (1999) showed that the partial resolvable n-cyclic designs with $r < k$ are not generally recommended as some main effects will necessarily be estimated inefficiently; see Chapter 5 and Appendix C. They further showed that more efficient designs with $r < k$ could be obtained by dropping complete replicates from the full n-cyclic set. It was through this line of investigation that the family of α_n-designs was ultimately discovered.

1.3 Assessing designs

Consider a factorial experiment with n factors where the ith factor has v_i levels and $v = v_1 v_2 \ldots v_n$ $(i = 1, 2 \ldots, n)$. Let the experiment be arranged in a resolvable block design with r replicates each with s blocks of size k, such that each treatment combination occurs once in each replicate. Then, the intrablock model is given by

$$y_{i_1 i_2 \ldots i_n;pj} = \mu + \rho_p + \beta_{pj} + \tau_{i_1 i_2 \ldots i_n} + \epsilon_{i_1 i_2 \ldots i_n;pj} \tag{1.1}$$

$$(i_m = 0, 1, \ldots, v_m - 1; m = 1, 2, \ldots, n; p = 1, 2 \ldots, r; j = 1, 2, \ldots, s)$$

where $y_{i_1 i_2 \ldots i_n;pj}$ is the yield of the plot in the jth block of the pth replicate to which the treatment combination $i_1 i_2 \ldots i_n$ has been applied, μ is the overall mean yield, $\tau_{i_1 i_2 \ldots i_n}$ is the fixed effect of the treatment combination $i_1 i_2 \ldots i_n$, ρ_p is the fixed effect of the pth replicate, β_{pj} is the fixed effect of the jth block in the pth replicate and $\epsilon_{i_1 i_2 \ldots i_n;pj}$ are independent and normally distributed random variables with means zero and homogeneous variances σ^2. (Definitions of the terminology used in the above discussion can be found in any text on the design of experiments; see, for example, John and Williams (1995, pp. 155–6) and Kuehl (2000, pp. 148,177–8).)

Under the model defined in (1.1) a solution to the normal equations is given by

$$\hat{\tau} = A^- q \tag{1.2}$$

where $\hat{\tau}$ is the least squares estimator of the $v \times 1$ vector τ whose elements are the treatment parameters $\tau_{i_1 i_2 \ldots i_n}$ defined in (1.1), and q is a vector of block adjusted treatment totals. The matrix A^- is any generalized inverse of the $v \times v$ *information matrix*

$$A = rI_v - NN'/k \tag{1.3}$$

where N is the treatment incidence matrix and NN' the treatment concurrence matrix of the design. Since A^- is a generalized inverse of A it satisfies $A = AA^-A$. For a more detailed discussion of the least-squares analysis of the intra-block model see John and Williams (1995, pp. 7–14).

For connected designs, in which all factorial effects are estimable, the rank of the information matrix is $v - 1$. Since the information matrix is not of full rank the

treatment parameters in τ are not uniquely estimable. However, if for any given value of m an $m \times v$ contrast matrix C, such that $C'1_v = 0$, satisfies the necessary and sufficient *estimability condition*

$$C = CA^{-}A \tag{1.4}$$

then a unique unbiased estimator of the contrasts of the treatment parameters, $C\tau$, is given by $C\hat{\tau}$ (Searle, 1971). Furthermore, it follows from the Gauss-Markov theorem that $C\hat{\tau}$ is also the best linear unbiased estimator of $C\tau$; see Dean and Voss (1999, p. 37). The variance-covariance matrix of $C\hat{\tau}$ is given by

$$\text{var}(C\hat{\tau}) = CA^{-}C'\sigma^2 \tag{1.5}$$

In a complete block design treatment effects are orthogonal to blocks so that all the information on treatment contrasts is available within blocks. However, for incomplete block designs only part of this information is available within blocks; the remainder must be obtained from treatment comparisons between blocks. An optimal design, therefore, is one whose configuration of treatments is such that it maximises the amount of information on within-block treatment contrasts. Thus, an objective function that discriminates between competing designs is needed to choose between different designs.

One choice of objective function is the *average efficiency factor*. It assesses the amount of within-block information on treatment contrasts by comparing the average within-block contrast variance for the design with the average contrast variance for a complete block design with the same replication per treatment, assuming the same error variance σ^2. Hence, the average efficiency factor is obtained by comparing the average variances in (1.5) with the variances in a complete block design, i.e.

$$E = \frac{\text{trace}(CC')}{r\,\text{trace}(CA^{-}C')} \tag{1.6}$$

since I_v/r is a generalized inverse of the information matrix for a complete block design and C is a contrast matrix satisfying the estimability condition given in (1.4). For the special case where $m = v$ and C is symmetric (i.e. $C' = C$) and idempotent

(i.e. $C^2 = C$) with $\text{rank}(C) = v - 1$, equation (1.6) simplifies to

$$E = \frac{v-1}{r\text{trace}(CA^-)} \tag{1.7}$$

since the trace of matrix products is invariant under any cyclic permutation of the order in which the matrices appear.

A detailed discussion of average efficiency factors can be found in John and Williams (1995, pp. 28–31).

1.4 Orthogonal factorial structure

A design has orthogonal factorial structure (OFS) if any treatment contrast belonging to one factorial effect, or generalized interaction (i.e. main effect or interaction), can be estimated independently of any other effect after the elimination of blocking effects. Hence, a design admitting OFS allows a straightforward analysis and interpretation of results. This is particularly desirable to the experimenter who is interested in drawing conclusions about contrasts belonging to different factorial effects.

John and Smith (1972) established a sufficient condition under which a two-factor experiment arranged in an incomplete block design has OFS. This condition was given in terms of a generalized inverse of the information matrix for the design. Cotter, John and Smith (1973) then extended this result to n-factor experiments. Mukerjee (1979) obtained a necessary and sufficient condition for OFS directly in terms of the information matrix. These conditions are given by Theorem 1.1; for a proof see Mukerjee (1979).

Before presenting Theorem 1.1 some notation and definitions are required. First, let any generalized interaction be represented by $F^x = F_1^{x_1} F_2^{x_2} \ldots F_n^{x_n}$, where $x = x_1 x_2 \ldots x_n$ and $x_i = 1$ if factor F_i is present in the interaction and $x_i = 0$ otherwise $(i = 1, 2, \ldots, n)$. The general mean $F^{00\ldots0}$ is nonestimable in any incomplete block design. Further, let C_x be a contrast matrix satisfying the estimability condition given in (1.4) and belonging to the factorial space for the n-factor interaction F^x. The *factorial space* for F^x, for any $x \neq 0$, is the set of $\nu_x = \prod_{i=1}^{n}(v_i - 1)^{x_i}$ linear functions $C_x\tau$ such that these linear functions are:

i) contrasts,

ii) belong to the interaction F^x, and

iii) account for all the degrees of freedom, ν_x, belonging to F^x.

Theorem 1.1 *For a connected block design, OFS holds if and only if the information matrix, A, and C_x commute for all $x \neq 0$, i.e. $C_x A = A C_x$.*

Some α_n-designs do not have OFS. This does not, however, preclude them from being useful in practice. While OFS is a desirable feature because it allows a straightforward analysis and interpretation of results, such designs exist for limited combinations of design parameters. If a design with OFS cannot be found, an α_n-design will provide a useful alternative. This is analogous to the situation where ideally a balanced incomplete block design is required but is unavailable for the given set of design parameters. In this case some structure would have to be sacrificed, and an unbalanced design such as an α-design could be used.

1.5 An algorithm for generating factorial designs

Williams and John (1996) described a computer algorithm for generating resolvable block and row-column designs for factorial experiments. The algorithm sets out to ensure that each level of every treatment factor occurs as equally as possible in each block of the design, or in each row and column of a row-column design. The expectation then is that main effects, at least, are estimated with a high degree of precision. While the algorithm does often produce practically useful designs, it is not entirely satisfactory. The aim of the algorithm is intuitively reasonable, however some of the generated designs have relatively low main effect precision. It is even possible that a design will not allow a main effect to be estimated at all, i.e. the design is disconnected with respect to that main effect. Further, the precision with which interactions are estimated is not directly considered.

1.6 Thesis outline

The family of α-designs provide a flexible class of resolvable incomplete block designs for single factor experiments comprising v treatments arranged in s blocks of size k with each treatment replicated r times. The design parameters together with some simple cyclic group theory play a key role in their construction. In Chapter 2 it will be shown that α-designs are constructed using a method of cyclic substitution. An important consequence of this method is that the treatment concurrence matrix of the design is block-circulant. This structure can be exploited to derive a concise mathematical expression for the average efficiency factor of an α-design and leads to substantial computational savings when used to assess competing designs. This chapter also serves as an introduction to the terminology and notation used in the remainder of the thesis.

In Chapter 3 it will be shown that α_n-designs can be constructed for any combination of design parameters that satisfy the factorisation $v = v_1 v_2 \dots v_n$ and $k = k_1 k_2 \dots k_n$ such that $v_i/k_i \in \mathbb{Z}^+$ for $i = 1, 2, \dots, n$. When these constraints are satisfied, the method used to construct α-designs naturally extends to α_n-designs. It will further be shown that all the properties of α-designs discussed in Chapter 2 also extend to α_n-designs.

Chapter 4 considers α_n-designs in the context of factorial experiments. Finding efficient designs for factorial experiments, particularly experiments involving a large number of factorial treatment combinations, requires an enormous amount of computational effort. However, a set of contrast matrices will be defined which facilitate the derivation of a closed form expression for the average efficiency factor of a generalized interaction. This expression will be shown to have a similar form to that of the average efficiency factor of an α-design, and will again lead to substantial computational savings when used to assess designs. Although α_n-designs do not, in general, admit orthogonal factorial structure it will be shown that they provide a flexible class of resolvable block designs for experiments with n factors.

The class of n-cyclic designs are a family of non-resolvable block designs that are particularly well-suited to factorial experiments. In Chapter 5 it will be shown

that there exists a subset of n-cyclic designs that are resolvable. It will further be shown that partial n-cyclic sets with $r < k$ can be constructed but that these are generally inefficient. It will further be shown that all resolvable n-cyclic designs are in fact α_n-designs. Designs with r equal to a multiple of k have orthogonal factorial structure.

This study has generated a number of questions. For instance, the current knowledge of the relationship between α_n-designs and resolvable n-cyclic designs requires further development. Moreover, the development of α_n-designs has suggested some new lines of research such as exploring the application and suitability of α_n-designs for experiments involving more complex treatment and blocking structures. These issues are discussed in Chapter 6.

Chapter 2

α-Designs

2.1 Introduction

Introduced by Patterson and Williams in 1976, α-designs were primarily developed for 'variety trials where the number of varieties, v, is often large but the number of replicates of each variety, r, is usually small', (Williams, 1975, p.1). They are now used worldwide in variety and other trials involving a single set of treatments in which a resolvable blocking structure is either necessary or desirable. In this chapter the construction and properties of α-designs are considered in detail so that the terminology and notation can later be extended to the development of α_n-designs.

2.2 Design construction

The construction of an α-design begins with a $k \times r$ *generating array* α whose elements $\alpha(p,q)$ are from the set $Z_s = \{0, 1, \ldots, s-1\}$ $(p = 1, 2, \ldots, k; q = 1, 2, \ldots, r)$. The set Z_s forms a group closed under addition modulo s, and is also known as the class of residues modulo s.

Cyclic substitution is carried out on each column of α by adding one to the elements in each column and reducing modulo s when necessary. In this way, a further $s-1$ columns are generated from each column of α. The resulting $k \times rs$ array, whose elements are also in the set Z_s, is called the *intermediate array* α^*.

The α-design is obtained from the intermediate array by adding the ith ordered element in the set $< s >= \{0, s, \ldots, (k-1)s\}$ to each element in the ith row of α^* $(i = 1, 2, \ldots, k)$. To illustrate, consider the following example.

Example 2.1 *Consider an experiment with $v = 24$ treatments arranged in a resolvable block design with block size $k = 4$, $s = 6$ blocks per replicate and $r = 3$ replicates. One possible generating array α is given in Table 2.1. The elements of α are from the set $Z_6 = \{0, 1, \ldots, 5\}$.*

Table 2.1: Generating array for $v = 24$, $k = 4$ and $r = 3$

α-array		
0	0	0
0	2	3
0	3	1
0	5	0

Carrying out cyclic substitution on each column of α yields the intermediate array α^* shown in Table 2.2. Notice that a further five columns are generated from each column of α.

Table 2.2: Intermediate array generated from the α-array in Table 2.1

α^*																		
0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
0	1	2	3	4	5	2	3	4	5	0	1	3	4	5	0	1	2
0	1	2	3	4	5	3	4	5	0	1	2	1	2	3	4	5	0
0	1	2	3	4	5	5	0	1	2	3	4	0	1	2	3	4	5

The α-design itself, given in Table 2.3, is generated by adding the ith ordered element in the set $< 6 >= \{0, 6, 12, 18\}$ to each element in the ith row of α^* $(i = 1, 2, 3, 4)$. Columns in the design represent blocks, and each group of six adjacent blocks comprises a complete replicate of the twenty-four treatments, as required.

Notice that the set of treatments $S_1 = \{0, 1, \ldots, 5\}$ occur exactly once in the first row of each replicate of the α-design in Table 2.3. Similarly, treatments in the sets

Table 2.3: α-Design obtained from the α-array in Table 2.1

Replicate 1						Replicate 2						Replicate 3					
0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
6	7	8	9	10	11	8	9	10	11	6	7	9	10	11	6	7	8
12	13	14	15	16	17	15	16	17	12	13	14	13	14	15	16	17	12
18	19	20	21	22	23	23	18	19	20	21	22	18	19	20	21	22	23

$S_2 = \{6, 7, \ldots, 11\}$, $S_3 = \{12, 13, \ldots, 17\}$ and $S_4 = \{18, 19, \ldots, 23\}$ occur exactly once in the second, third and last rows, respectively, of each replicate. Consequently, each block contains exactly one treatment from each of S_1, S_2, S_3 and S_4. These four sets partition the treatment labels in Z_{24} into mutually disjoint subsets, thereby ensuring the resolvable structure of the α-design. The first subset, S_1, is just the cyclic group Z_6. The remaining three subsets are generated by adding the ith ordered element of the subgroup $< 6 >$ to each element in Z_6 $(i = 2, 3, 4)$.

In general, an α-design can be constructed for $v = ks$ treatments where k is the block size and s is the number of blocks per replicate. Since k divides v, the set of v treatment labels $Z_v = \{0, 1, \ldots, v-1\}$ can be partitioned into k mutually disjoint subsets of order s. The first subset, S_1, is always the set Z_s. The remaining $k-1$ subsets are obtained by adding the ith ordered element, t_i, in $< s >$ to each of the elements in Z_s $(i = 2, 3, \ldots, k)$, i.e. $S_i = Z_s + t_i = \{t_i, 1 + t_i, \ldots, s - 1 + t_i\}$.

2.3 The treatment concurrence matrix, NN'

Let the rows of the treatment incidence matrix, N, correspond to the ordered treatment labels in Z_v and let the columns of N correspond to the jth block in the design $(j = 1, 2, \ldots, b)$. Then, the treatment concurrence matrix, NN', of the α-design in Table 2.3 is given by the 24×24 matrix in Figure 2.1. Its rows and columns are labelled with respect to the ordered treatments in Z_{24} and is partitioned into 4^2, 6×6 submatrices $M_{\ell m}$ $(\ell, m = 1, 2, 3, 4)$. The (i, j)th element of any submatrix $M_{\ell m}$ is the number of blocks in which the treatments $a_{6(\ell-1)+i-1} \in S_\ell$ and $b_{6(m-1)+j-1} \in S_m$ concur $(i, j = 1, 2, \ldots, 6)$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	3	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	0	0	2	0	0	0	0	1
1	0	3	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	0	1	2	0	0	0	0
2	0	0	3	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	0	1	2	0	0	0
3	0	0	0	3	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	0	1	2	0	0
4	0	0	0	0	3	0	1	1	0	0	1	0	0	1	0	0	1	1	0	0	0	1	2	0
5	0	0	0	0	0	3	0	1	1	0	0	1	1	0	1	0	0	1	0	0	0	0	1	2
6	1	0	0	1	1	0	3	0	0	0	0	0	1	1	0	0	1	0	1	0	0	2	0	0
7	0	1	0	0	1	1	0	3	0	0	0	0	0	1	1	0	0	1	0	1	0	0	2	0
8	1	0	1	0	0	1	0	0	3	0	0	0	1	0	1	1	0	0	0	0	1	0	0	2
9	1	1	0	1	0	0	0	0	0	3	0	0	0	1	0	1	1	0	2	0	0	1	0	0
10	0	1	1	0	1	0	0	0	0	0	3	0	0	0	1	0	1	1	0	2	0	0	1	0
11	0	0	1	1	0	1	0	0	0	0	0	3	1	0	0	1	0	1	0	0	2	0	0	1
12	1	0	0	1	0	1	1	0	1	0	0	1	3	0	0	0	0	0	1	0	1	0	0	1
13	1	1	0	0	1	0	1	1	0	1	0	0	0	3	0	0	0	0	1	1	0	1	0	0
14	0	1	1	0	0	1	0	1	1	0	1	0	0	0	3	0	0	0	0	1	1	0	1	0
15	1	0	1	1	0	0	0	0	1	1	0	1	0	0	0	3	0	0	0	0	1	1	0	1
16	0	1	0	1	1	0	1	0	0	1	1	0	0	0	0	0	3	0	1	0	0	1	1	0
17	0	0	1	0	1	1	0	1	0	0	1	1	0	0	0	0	0	3	0	1	0	0	1	1
18	2	1	0	0	0	0	1	0	0	2	0	0	1	1	0	0	1	0	3	0	0	0	0	0
19	0	2	1	0	0	0	0	1	0	0	2	0	0	1	1	0	0	1	0	3	0	0	0	0
20	0	0	2	1	0	0	0	0	1	0	0	2	1	0	1	1	0	0	0	0	3	0	0	0
21	0	0	0	2	1	0	2	0	0	1	0	0	0	1	0	1	1	0	0	0	0	3	0	0
22	0	0	0	0	2	1	0	2	0	0	1	0	0	0	1	0	1	1	0	0	0	0	3	0
23	1	0	0	0	0	2	0	0	2	0	0	1	1	0	0	1	0	1	0	0	0	0	0	3

Figure 2.1: Treatment concurrence matrix for the α-design in Table 2.3

A submatrix $M_{\ell\ell}$ along the leading diagonal of NN' contains the within block concurrences of pairs of treatments from the same subset S_ℓ. Since the method of construction prevents the concurrence of two distinct treatments from the same subset, all the off-diagonal elements of $M_{\ell\ell}$ are zero. The concurrences along the leading diagonal of each such submatrix are 3. This is because an element along the leading diagonal is the number of concurrences of a treatment with itself, i.e. the number of replications of that treatment. Hence, $M_{\ell\ell} = 3I_6$ for $\ell = 1, 2, 3, 4$. In general, $M_{\ell m} = rI_s$ for all $\ell = m$ $(\ell, m = 1, 2, \ldots, k)$.

Now consider the off-diagonal submatrices of the treatment concurrence matrix

in Figure 2.1. In particular, consider the submatrix M_{12}, i.e.

$$M_{12} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

The (i,j)th element in M_{12} is the number of concurrences of the ith and jth ordered treatments in subsets S_1 and S_2 respectively. For example, the element in the first row and third column is 1 and indicates that treatments 0 and 8 concur in one block. Further, the method of cyclic substitution is such that if treatments 0 and 8 concur in a block so too do treatments 1 and 9, 2 and 10, ...,5 and 7. Notice that the submatrix M_{12} is a circulant matrix, i.e. in each successive row the elements move to the right one position (with wrap around at the edges).

More generally, consider the two treatments $a_\ell \in S_\ell$ and $b_m \in S_m$ for $\ell, m = 1, 2, \ldots, k$ and $\ell \neq m$. These treatments can be factorised as $a_\ell = (\ell - 1)s + p$ and $b_m = (m-1)s + q$, respectively, where p and q are elements of Z_s. It follows that the difference between a_ℓ and b_m, reducing modulo s, is given by

$$h = (q - p) \text{ modulo } s = \begin{cases} q - p & \text{if } q \geq p \\ s - |q - p| & \text{if } q < p \end{cases} \tag{2.1}$$

where $h \in Z_s$. Now, for $\ell \leq m$ let $b_m - a_\ell$ be defined as in (2.1) and let λ_h denote the number of blocks in which treatments a_ℓ and b_m concur. Then, the submatrix $M_{\ell m}$ can be written as

$$M_{\ell m} = \begin{pmatrix} \lambda_0 & \lambda_1 & \cdots & \lambda_{s-1} \\ \lambda_{s-1} & \lambda_0 & \cdots & \lambda_{s-2} \\ \vdots & & & \\ \lambda_1 & \lambda_2 & \cdots & \lambda_0 \end{pmatrix}$$

where λ_h is the number of concurrences of $a_\ell \in S_\ell$ and $b_m \in S_m$ whose difference, $b_m - a_\ell$, is h after reducing modulo s. Hence,

$$M_{\ell m} = \sum_{h=0}^{s-1} \lambda_h \otimes \Gamma_h \tag{2.2}$$

where Γ_h is an $s \times s$ *basic* circulant matrix; that is, its first row has a one in the $(h+1)$th column and all other elements zero. Successive rows of Γ_h are obtained by

moving the elements in the previous row to the right one position (with wrap-around at the end). From (2.1), if $\ell > m$ then

$$M_{m\ell} = \sum_{h=0}^{s-1} \lambda_{s-h} \otimes \Gamma_h \tag{2.3}$$

Since $\Gamma_h' = \Gamma_{s-h}$, it follows that $M_{m\ell} = M_{\ell m}'$.

Now, the treatment concurrence matrix in Figure 2.1 can be written as

$$NN' = \begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{pmatrix}$$

From (2.2) and (2.3) it follows that the entire treatment concurrence matrix can be specified by the elements in the first row of each submatrix $M_{\ell m}$ so that

$$NN' = \sum_{h=0}^{5} B_h \otimes \Gamma_h \tag{2.4}$$

where

$$B_h = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ \beta_{21} & \beta_{22} & \beta_{23} & \beta_{24} \\ \beta_{31} & \beta_{32} & \beta_{33} & \beta_{34} \\ \beta_{41} & \beta_{42} & \beta_{43} & \beta_{44} \end{pmatrix}$$

and $\beta_{\ell m}$ is the concurrence λ_h in the submatrix $M_{\ell m}$ ($h = 0, 1, \ldots, 5; \ell, m = 1, 2, 3, 4$). Hence, the six 4×4 matrices B_h obtained from NN' in Figure 2.1 are

$$B_0 = \begin{pmatrix} 3 & 1 & 1 & 2 \\ 1 & 3 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 2 & 1 & 1 & 3 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad B_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B_3 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix} \quad B_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B_5 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Since NN' is symmetric and $\Gamma_h' = \Gamma_{s-h}$, then $B_0' = B_0$, $B_1' = B_5$, $B_2' = B_4$ and $B_3' = B_3$, i.e. $B_h' = B_{s-h}$.

In general, the treatment concurrence matrix of an α-design can be be partitioned into k^2 $s \times s$ submatrices $M_{\ell m}$, i.e.

$$NN' = \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1k} \\ M_{21} & M_{22} & \cdots & M_{2k} \\ \vdots & & & \\ M_{k1} & M_{k2} & \cdots & M_{kk} \end{pmatrix}$$

where the (i,j)th element of the submatrix $M_{\ell m}$ is the within block concurrence of the treatments $a_{(\ell-1)s+i-1} \in S_\ell$ and $b_{(m-1)s+j-1} \in S_m$ $(i,j = 1,2,\ldots,s;\ \ell,m = 1,2,\ldots,k)$. The submatrices along the leading diagonal of NN' are given by $M_{\ell\ell} = rI_s$ for all $\ell = 1,2,\ldots,k$, while the off-diagonal submatrices are given by

$$M_{\ell m} = \begin{pmatrix} \lambda_0 & \lambda_1 & \cdots & \lambda_{s-1} \\ \lambda_{s-1} & \lambda_0 & \cdots & \lambda_{s-2} \\ \vdots & & & \\ \lambda_1 & \lambda_2 & \cdots & \lambda_0 \end{pmatrix}$$

where λ_h is the number of concurrences of $a_\ell \in S_\ell$ and $b_m \in S_m$ whose difference, $b_m - a_\ell$, is h after reducing modulo s. Hence, the treatment concurrence matrix of an α-design is given by

$$NN' = \sum_{h=0}^{s-1} B_h \otimes \Gamma_h \tag{2.5}$$

where B_h is a $k \times k$ matrix whose (ℓ,m)th element is the concurrence λ_h in $M_{\ell m}$, and $B'_h = B_{s-h}$. Since each submatrix $M_{\ell m}$ has circulant structure, or C-structure, the treatment concurrence matrix is said to have *block circulant structure*, or *BC-structure.*

2.4 Generating NN' from an α-array

In this section it will be shown that the treatment concurrence matrix of any α-design can be obtained directly from its generating array α. This means that construction of the full design is unnecessary. Furthermore, computer algorithms for the generation of efficient α-designs require fewer operations since competing designs

are obtained by exchanging elements in the generating array with suitable candidate elements from the set Z_s.

From the method of construction described in section 2.2, treatments in the ℓth row of an α-design are obtained by adding the constant $(\ell-1)s$ to each element in the ℓth row of the intermediate array α^* $(\ell = 1, 2, \ldots, k)$. Therefore, a treatment a_ℓ occurring in the ℓth row of an α-design can be factorised as $a_\ell = (\ell-1)s+p$, where $p \in Z_s$. For example, consider the α-design in Table 2.3. The pair of treatments 8 and 15 which concur in the first block of replicate 2 can be factorised as $8 = 1\times 6+p_1$ and $15 = 2 \times 6 + q_1$, where $p_1 = 2$ and $q_1 = 3$. The values of p_1 and q_1 are the elements in the corresponding rows and column of the intermediate array α^* in Table 2.2.

Now, consider the intermediate array α^* in Table 2.2. It consists of three sets of six adjacent columns where the initial column of the jth set is the jth column of the generating array α in Table 2.1 $(j = 1, 2, 3)$. The columns within a set are generated successively by adding 1 to the elements in the previous column, reducing modulo 6 when necessary. If p_t and q_t represent the elements in the ℓth and mth rows, respectively, of the tth column within a set then $(q_t - p_t)$ modulo 6 is constant over all t $(\ell, m = 1, 2, 3, 4;\ t = 1, 2, \ldots, 6)$. For example, again consider the treatments 8 and 15 in the first block of replicate 2 in Table 2.3. The difference $(q_1 - p_1)$ modulo 6 is 1. Further, $(q_t - p_t)$ modulo 6 is 1 for $t = 2, 3, \ldots, 6$. Since $8 \in S_2$ and $15 \in S_3$ it follows that all pairs of treatments from S_2 and S_3 whose difference is 1, after reducing modulo 6, concur within the blocks of replicate 2. These treatment concurrences could have been obtained directly from the second column of the generating array α in Table 2.1.

More generally, let $\alpha(\ell, j)$ represent the (ℓ, j)th element of the generating array α. For example, consider the generating array α in Table 2.1. Then for $j = 1, 2$ and 3 the differences $\alpha(3, j) - \alpha(2, j)$ are 0, 1 and 4, respectively, after reducing modulo 6. It follows that those pairs of treatments in S_2 and S_3 whose difference is 0, 1 or 4, after reducing modulo 6, concur once throughout the design. From the definition of the $k \times k$ matrix B_h in (2.5) it follows that the element in the second row and third column of the matrices B_0, B_1 and B_4 is 1, and zero in B_2, B_3 and

B_5 $(h = 0, 1, \ldots, 5)$. Doing this for $\ell < m$ yields only the concurrences in the upper triangle of each B_h matrix. The lower triangle of each of these matrices is readily obtained by using the property $B_h' = B_{s-h}$. For example, since $B_1' = B_5$ it follows that β_{21} in B_1 equals β_{12} in B_5, and more generally, $\beta_{\ell m}$ in B_h equals $\beta_{m\ell}$ in B_{s-h}. Finally, consider the diagonal elements $\beta_{\ell\ell}$ of the B_h matrices $(\ell = 1, 2, 3, 4)$. For $h = 0$, $\beta_{\ell\ell}$ is the number of concurrences of the first ordered treatment in the subset S_ℓ with itself. Hence, $\beta_{\ell\ell} = 3$ for all ℓ. For $h > 0$, $\beta_{\ell\ell}$ is the number of concurrences of the first ordered treatment in the subset S_ℓ with the $(h+1)$th ordered treatment in the subset S_ℓ so that $\beta_{\ell\ell} = 0$. Hence, the α-array is not needed for obtaining the diagonal elements of the B_h matrices.

2.5 The information matrix, A

The information matrix of a resolvable block design with constant block size is given by

$$A = rI_v - NN'/k$$

It has BC-structure because it is a simple linear function of the treatment concurrence matrix. Substituting the treatment concurrence matrix given in (2.5) into the above expression for A gives

$$A = \sum_{h=0}^{s-1} A_h \otimes \Gamma_h \tag{2.6}$$

where

$$A_h = \begin{cases} rI_k - B_h/k & h = 0 \\ -B_h/k & h > 0 \end{cases} \tag{2.7}$$

and where B_h is defined as in (2.4).

Following Davis and Hall (1969), the canonical form of the basic circulant matrix Γ_h of order s is given by

$$\Gamma_h = \sum_{u=0}^{s-1} \omega_u^h \Gamma_u^* \tag{2.8}$$

where

$$\Gamma_u^* = \gamma_u \overline{\gamma}_u' \tag{2.9}$$

and where

$$\gamma_u' = \left(\omega_u^0, \omega_u^1, \omega_u^2, \ldots, \omega_u^{s-1}\right)/\sqrt{s} \tag{2.10}$$

is a normalised eigenvector of Γ_h with corresponding eigenvalue ω_u^h, where $\omega_u = \exp(2\pi iu/s)$ is the uth of the sth roots of unity $(h, u = 0, 1, \ldots, s-1)$; see Theorem B.3 in Appendix B. Substituting (2.8) into (2.6) gives

$$A = \sum_{u=0}^{s-1} A_u^* \otimes \Gamma_u^* \tag{2.11}$$

where

$$A_u^* = \sum_{h=0}^{s-1} \omega_u^h A_h \tag{2.12}$$

Note from (2.9) and (2.10) the matrices Γ_u^* can be written as a weighted average of the basic circulant matrices Γ_h, i.e.

$$\Gamma_u^* = \frac{1}{s} \sum_{h=0}^{s-1} \omega_u^{-h} \Gamma_h \tag{2.13}$$

These matrices are mutually orthogonal and idempotent since

$$\gamma_t' \bar{\gamma}_u = \begin{cases} 1 & \text{if } t = u \\ 0 & \text{if } t \neq u \end{cases}$$

$(u, t = 0, 1, \ldots, s-1)$. Equation (2.9) gives the canonical form of the Γ_u^* matrices. It shows that they each have a single non-zero eigenvalue, namely unity, with corresponding eigenvector γ_u. Hence, $\text{rank}(\Gamma_u^*) = \text{trace}(\Gamma_u^*) = 1$. These properties of Γ_u^* are useful in obtaining a generalized inverse of A; see section 2.5.2.

2.5.1 Properties of A

For a connected design, $\text{rank}(A) = v - 1$. Hence, A has $v - 1$ non-zero eigenvalues. It will now be shown that when the information matrix is defined as in (2.11) the singularity in A is contained in the $k \times k$ matrix A_0^*. From (2.12) and (2.7) it follows that $A_0^* = r(I_k - K_k)$, where the averaging matrix $K_k = 1_k 1_k'/k$ and 1_k is a $k \times 1$ vector of ones. Since the matrix K_k is symmetric and idempotent then $\text{rank}(A_0^*) = \text{rank}(I_k) - \text{rank}(K_k) = k - 1$, i.e. the zero eigenvalue of A is from the system of equations given by A_0^*; see Theorem B.2 in Appendix B. Hence, the

remaining $s-1$ A_u^* matrices each have k non-zero eigenvalues, i.e. they are each of full rank ($u = 1, 2, \ldots, s-1$).

Another property of the information matrix defined in (2.11) is that the $k \times k$ matrices A_u^* are Hermitian, i.e. $(A_u^*)^\dagger = A_u^*$ where $(A_u^*)^\dagger$ is the conjugate transpose of A_u^*; see Definition B.6 in Appendix B. This result is used in section 2.6 to show that complex matrix inversion can be avoided when computing the average efficiency factor, E, of an α-design.

2.5.2 A generalized inverse of A

In this section a generalized inverse of the information matrix of an α-design, defined in (2.11), is given. This generalized inverse follows directly from Theorem 2.1 below.

Theorem 2.1 *Let A_u^* be any $k \times k$ matrix and let A_u^{*-} be a generalized inverse of A_u^*, i.e. $A_u^* A_u^{*-} A_u^* = A_u^*$. Further, let the $s \times s$ matrices Γ_u^* and Γ_t^* be mutually orthogonal (i.e. $\Gamma_u^* \Gamma_t^* = 0$ for $u \neq t$) and idempotent (i.e. $\Gamma_u^{*2} = \Gamma_u^*$). Then, with $v = ks$, the $v \times v$ matrix*

$$A^- = \sum_{u=0}^{s-1} A_u^{*-} \otimes \Gamma_u^*$$

is a generalized inverse of the matrix

$$A = \sum_{u=0}^{s-1} A_u^* \otimes \Gamma_u^*$$

i.e. $AA^-A = A$.

Proof. Let A_u^{*-} be a generalized inverse of the $k \times k$ matrix A_u^* and let the $s \times s$ matrices Γ_u^* and Γ_t^* be mutually orthogonal and idempotent. Now, consider the $v \times v$ matrix

$$A = \sum_{u=0}^{s-1} A_u^* \otimes \Gamma_u^*$$

Then, by Theorem B.6 in Appendix B

$$AA^-A = \sum_{u=0}^{s-1} \sum_{x=0}^{s-1} \sum_{y=0}^{s-1} A_u^* A_x^{*-} A_y^* \otimes \Gamma_u^* \Gamma_x^* \Gamma_y^*$$

Since Γ_u^*, Γ_x^* and Γ_y^* are mutually orthogonal and idempotent, and A_u^{*-} is a generalized inverse of A_u^*, it follows that

$$AA^-A = \sum_{u=0}^{s-1} A_u^* A_u^{*-} A_u^* \otimes \Gamma_u^* = A$$

Hence, A^- is a generalized inverse of A.

End of proof.

A generalized inverse of A_0^* is $(A_0^* + rK_k)^{-1}$ (Shah, 1959). Since $A_0^* = r(I_k - K_k)$ then $A_0^{*-} = I_k/r$. Since A_u^* is non-singular for $u > 0$, a generalized inverse of the information matrix of an α-design is given by

$$A^- = \sum_{u=0}^{s-1} A_u^{*-} \otimes \Gamma_u^*$$

where

$$A_u^{*-} = \begin{cases} I_k/r & u = 0 \\ A_u^{*-1} & u > 0 \end{cases} \tag{2.14}$$

The information matrix plays an integral role in the calculation of the average efficiency factor, E, used in comparing alternative designs of the same size. Hence, the motivation for using the definitions of Davis and Hall (1969) does not lie in deriving elegant mathematical expressions for A, in (2.11), and A^-, in (2.14), but in the potential gains in computational efficiency when calculating E. It will be seen in the following section that these definitions for A and A^- lead to computationally cheaper methods of calculating E by reducing the problem of inverting a $v \times v$ matrix to one of inverting $s-1$ $k \times k$ matrices, where $v > k$. Further savings are made by using the Hermitian properties of A_u^* to avoid complex matrix inversion in obtaining A_u^{*-}.

2.6 The average efficiency factor, E

In chapter 1 the average efficiency factor, E, was described as a measure, or objective function, used to compare different designs of the same size (v, k, r). In section 1.3 it was shown that for a symmetric and idempotent contrast matrix, C, of rank $v-1$ the average efficiency factor is given by (1.7), namely

$$E = \frac{v-1}{r\mathrm{tr}(CA^-)} \tag{2.15}$$

One choice for C is the matrix $I_v - K_v$ which can be written as

$$C = I_k \otimes I_s - K_k \otimes K_s$$

so that

$$CA^- = \sum_{u=0}^{s-1} (I_k A_u^{*-} \otimes I_s \Gamma_u^*) - (K_k A_u^{*-} \otimes K_s \Gamma_u^*) \tag{2.16}$$

The following lemma is required in simplifying (2.16) and hence, in proving Lemma 2.2.

Lemma 2.1 *Let the $s \times s$ matrix L_s denote either the identity matrix I_s or the averaging matrix K_s. Also, let the $s \times s$ matrix Γ_u^* be defined as in (2.13). Then*

$$L_s \Gamma_u^* = \begin{cases} K_s & \text{if } u = 0 \\ \Gamma_u^* & \text{if } u > 0 \text{ and } L_s = I_s \\ 0_s & \text{if } u > 0 \text{ and } L_s = K_s \end{cases} \tag{2.17}$$

Proof. The first two results are trivial to show. From (2.13), $\Gamma_0^* = K_s$ and is therefore idempotent. Hence, $I_s \Gamma_0^* = K_s \Gamma_0^* = K_s$. When $u > 0$, $I_k \Gamma_u^* = \Gamma_u^*$. However,

$$\begin{aligned} K_s \Gamma_u^* &= \frac{1}{s^2} \sum_{h=0}^{s-1} \omega_u^{-h} J_s \Gamma_h \\ &= \frac{1}{s^2} \sum_{h=0}^{s-1} \omega_u^{-h} J_s \end{aligned}$$

where $J_s = 1_s 1_s'$. It follows that every element of $K_s \Gamma_u^*$ equals $(1/s^2) \sum_{h=0}^{s-1} \omega_u^{-h}$. Using the geometric series identity, and since $\omega_u^{-h} = \omega_u^{s-h}$, it follows that

$$\sum_{h=0}^{s-1} \omega_u^{-h} = \frac{1 - \omega_u^s}{1 - \omega_u} = 0$$

Hence, for $u > 0$ every element of $K_s \Gamma_u^*$ is zero, yielding the $s \times s$ zero matrix 0_s. *End of proof.*

Lemma 2.2 *For α-designs with r replicates of v treatments arranged in blocks of size k, the average efficiency factor is given by*

$$E = \frac{v - 1}{(k-1) + r \sum_{u=1}^{s-1} trace(A_u^{*-1})} \tag{2.18}$$

where the $k \times k$ matrix A_u^* is defined as in (2.12).

Proof. Using Lemma 2.1, the matrix product CA^{-} given in (2.16) can be partitioned into the contributions $[(I_k - K_k) \otimes K_s]/r$ for $u = 0$, and $\sum_{u=1}^{s-1} A_u^{*-1} \otimes \Gamma_u^*$ for $u > 0$, i.e.

$$CA^{-} = \frac{1}{r}(I_k - K_k) \otimes K_s + \sum_{u=1}^{s-1} A_u^{*-1} \otimes \Gamma_u^*$$

Thus,

$$\text{trace}(CA^{-}) = \frac{1}{r}(k-1) + \sum_{u=1}^{s-1} \text{trace}(A_u^{*-1}) \tag{2.19}$$

since $\text{trace}(\Gamma_u^*) = 1$. Substituting (2.19) into (2.15) yields the expression for E given in (2.18). Note that the constant $k - 1$ in the denominator of (2.18) is the contribution from A_0^{*-} and equals its rank.

End of proof.

Since the $k \times k$ matrix A_u^* in (2.18) is Hermitian, Corollary 2.1 follows directly from Lemma 2.2.

Corollary 2.1 *Calculation of the average efficiency factor, E, defined in (2.18) does not require complex matrix inversion, i.e.*

$$E = \frac{v-1}{(k-1) + r\sum_{u=1}^{s-1} \textit{trace}\left[(X_{1u} + X_{2u}X_{1u}^{-1}X_{2u})^{-1}\right]} \tag{2.20}$$

where the $k \times k$ matrices X_{1u} and X_{2u}, with real entries, are the real and imaginary parts of A_u^*.

Proof. Since the matrix A_u^* is Hermitian, A_u^{*-1} is also Hermitian; see section 2.5.1 and Theorem B.9 in Appendix B. Thus, A_u^{*-1} can be written in terms of its real and imaginary parts, i.e. $A_u^{*-1} = Y_{1u} + iY_{2u}$, where Y_{ju} is a $k \times k$ matrix with real entries and $i^2 = -1$ $(j = 1, 2)$. It follows that Y_{2u} is skew-symmetric and hence, $\text{trace}(A_u^{*-1}) = \text{trace}(Y_{1u})$. Writing $A_u^* = X_{1u} + iX_{2u}$, it now follows that

$$(X_{1u} + iX_{2u})\,(Y_{1u} + iY_{2u}) = I$$

Expanding the left hand side of this equation and equating the real part to I and the imaginary part to 0 yields

$$X_{1u}Y_{1u} - X_{2u}Y_{2u} \;=\; I \tag{2.21}$$

and

$$X_{1u}Y_{2u} + X_{2u}Y_{1u} = 0 \tag{2.22}$$

Equations (2.21) and (2.22) can now be solved for Y_{1u} and Y_{2u}. Although Y_{2u} is not needed in calculating E, its solution is also found for completeness.

Adding the two equations resulting from pre-multiplying (2.21) by X_{u2}^{-1} and (2.22) by X_{u1}^{-1} yields

$$\left(X_{2u}^{-1}X_{1u} + X_{1u}^{-1}X_{2u}\right) Y_{1u} = X_{2u}^{-1}$$

so that

$$Y_{1u} = \left(X_{1u} + X_{2u}X_{1u}^{-1}X_{2u}\right)^{-1} \tag{2.23}$$

Similarly, adding the two equations resulting from pre-multiplying (2.21) by $-X_{1u}^{-1}$ and (2.22) by X_{2u}^{-1} yields

$$\left(X_{1u}^{-1}X_{2u} + X_{2u}^{-1}X_{1u}\right) Y_{2u} = -X_{1u}^{-1}$$

so that

$$Y_{2u} = -\left(X_{2u} + X_{1u}X_{2u}^{-1}X_{1u}\right)^{-1} \tag{2.24}$$

Substituting A_u^{*-1} in (2.18) with Y_{1u} in (2.23) yields the result in (2.20). Hence, complex matrix inversion can be avoided when calculating E.

End of proof.

2.7 A design screening function

Constraining the search for an optimal resolvable incomplete block design to the family of α-designs reduces the number of designs that must be considered. This in itself results in savings in computing time. However, numerous designs must still be considered. For example, if the elements of each column of the $k \times r$ generating array α are chosen by fixing $\alpha(1, j) = 0$ and then randomly selecting the remaining $k - 1$ labels in each column from the set Z_s, with replacement, the number of possible permutations of the s labels for any column is $n_c = s^{k-1}$ $(j = 1, 2, \ldots, r)$. The replicates generated from these n_c columns are distinct. For an r replicate

experiment, therefore, there are ${}^{n_c}C_r$ ways in which r columns can be chosen from the n_c possible columns. Hence, for a variety trial with $v = 100$ varieties arranged in blocks of size $k = 10$ with each variety replicated $r = 3$ times, the number of candidate α-designs is in excess of one billion.

Computing the average efficiency factor, E, for every candidate α-design is computationally very intensive. Even with the considerable savings that are made by utilizing the inherent structure underlying α-designs, greater savings can be made by screening competing designs using a function based on a *counting rule* requiring no eigenvalue calculations. Then, E is calculated only when the value of the screening function for the current design exceeds that of the previous design.

One such counting rule is that based on the (M, S)-*optimality criterion* introduced by Shah (1960) and Eccleston and Hedayat (1974). It involves a two-stage process. First, a subclass of designs which maximimises trace(A) is formed. For an α-design, this quantity equals the constant $vr(k-1)/k$ and, therefore, does not discriminate between designs of the same size.

The second stage of the process involves finding the design, or designs, which minimise trace(A^2). For a resolvable incomplete block design

$$\text{trace}(A^2) = vr^2(k-1)^2/k^2 + \sum_{i \neq j} \lambda_{ij}^2/k^2 \tag{2.25}$$

where λ_{ij} is the number of blocks in which both treatments i and j occur. Note that for any resolvable block design $\lambda_{ii}^2 = r^2$ and for this reason its contribution to trace(A^2) is included in the constant term in (2.25). Further, since the constant term provides no additional screening information, it is sufficient to compute only the sum of squared off-diagonal elements of the treatment concurrence matrix. Hence, it now follows that minimising trace(A^2) is equivalent to minimising the sum of the squared off-diagonal elements of the treatment concurrence matrix. This is intuitively appealing since minimising $\sum_{i \neq j} \lambda_{ij}^2$ requires that a design has all treatment concurrences as equal as possible.

For resolvable block designs with constant block size

$$\sum_{i \neq j} \lambda_{ij}^2 = \text{trace}\left[(NN')^2\right] - vr^2$$

For α-designs

$$\begin{aligned}\text{trace}\left[(NN')^2\right] &= \text{trace}\left[\left(\sum_{h=0}^{s-1} B_h \otimes \Gamma_h\right)^2\right] \\ &= \sum_{g=0}^{s-1}\sum_{h=0}^{s-1} \text{trace}\,(B_g B_h)\,\text{trace}\,(\Gamma_{h_1}\Gamma_{h_2})\end{aligned}$$

where the $k \times k$ matrices B_g and B_h are defined as in (2.5). Now, since

$$\text{trace}(\Gamma_g \Gamma_h) = \begin{cases} s & \text{if } h = g - s \\ 0 & \text{otherwise} \end{cases}$$

and $B_g' = B_{s-g}$, it follows that

$$\text{trace}\left[(NN')^2\right] = \begin{cases} s\left\{\sum_{\forall \ell,m}[\beta_{\ell m}^2(0) + \beta_{\ell m}^2(a)] + 2\sum_{g=1}^{a-1}\sum_{\forall \ell,m}\beta_{\ell m}^2(g)\right\} & \text{if } s \text{ is even} \\ s\left[\sum_{\forall \ell,m}\beta_{\ell m}^2(0) + 2\sum_{g=1}^{a-1}\sum_{\forall \ell,m}\beta_{\ell m}^2(g)\right] & \text{if } s \text{ is odd} \end{cases}$$

where

$$a = \begin{cases} s/2 - 1 & \text{if } s \text{ is even} \\ (s-1)/2 & \text{if } s \text{ is odd} \end{cases} \tag{2.26}$$

and where $\sum_{\forall \ell,m}$ is the summation over all combinations of ℓ and m and $\beta_{\ell m}(g)$ is the (ℓ, m)th element of the $k \times k$ matrix B_g. Further note that $\text{trace}(B_0^2) = kr^2 + \sum_{\ell \neq m}\beta_{\ell m}^2(0)$, where kr^2 is the contribution to trace$[(NN')^2]$ from the sum of squared diagonal elements of NN'. Furthermore, the diagonal elements of each submatrix B_g are zero for $g > 0$. Hence, the sum of squared off-diagonal elements of NN' is given by

$$\sum_{i \neq j}\lambda_{ij}^2 = \begin{cases} s\left\{\sum_{\ell \neq m}[\beta_{\ell m}^2(0) + \beta_{\ell m}^2(a)] + 2\sum_{g=1}^{a-1}\sum_{\ell \neq m}\beta_{\ell m}^2(g)\right\} & \text{if } s \text{ is even} \\ s\left[\sum_{\ell \neq m}\beta_{\ell m}^2(0) + 2\sum_{g=1}^{a-1}\sum_{\ell \neq m}\beta_{\ell m}^2(g)\right] & \text{if } s \text{ is odd} \end{cases} \tag{2.27}$$

where $\sum_{\ell \neq m}$ is the summation over all combinations of ℓ and m except $\ell = m = 1$, and a is defined as in (2.26). Thus, the sum of squared off-diagonal elements of NN' is given by s times the sum of squared off-diagonal elements of the $k \times k$ matrices B_g for $g = 0, 1, \ldots, s-1$. Further savings are made by making use of the property $B_g' = B_{s-g}$, as shown by (2.27).

The (M, S)-optimality criterion provides a very efficient method of screening competing designs. However, further savings in computational effort are achieved by exploiting the BC-structure in the treatment concurrence matrix of α-designs. For instance, since the treatment concurrence matrix is symmetric only the concurrences λ_{ij} in either the upper or lower triangle of NN' must be squared. This involves $v(v-1)/2$ multiplications. However, by making use of the BC-structure of NN' for α-designs, as was done in deriving the screening function in (2.27), the number of multiplications reduces to $(k-1)(v+2k)/2$ if s is even and $(v+k)(k-1)/2$ if s is odd. The additional savings in computational effort are substantial. For example, in a variety trial with $v = 100$ varieties arranged in blocks of size $k = 10$ using the screening function in (2.27) reduces the number of multiplications by around 90%.

2.8 An alternative method of construction

In section 2.2 it was shown that α-designs are completely specified by a $k \times r$ generating array α whose elements are from the group Z_s closed under addition modulo s. Replicates were generated by cyclic development of each column of α yielding an intermediate array α^* and then adding the ith ordered element of the subgroup $< s >$ to each element in the ith row of α^* $(i = 1, \ldots, k)$.

An alternative approach, due to Jarrett and Hall (1978), is to partition the v treatments into the k mutually disjoint subsets

$$S = \{0, k, 2k, \ldots, (s-1)k\}$$

and

$$C_i = \{i, k+i, 2k+i, \ldots, (s-1)k+i\}$$

for $i = 1, 2, \ldots, k-1$. The subset S is simply the subgroup $< k >$, generated by k, of Z_v closed under addition modulo v. The remaining $k-1$ subsets, C_i, are the *cosets* of $< k >$. This approach is now considered and will be used later in section 5.2 to show the relationship that exists between α-designs and cyclic designs.

Construction, using this alternative partitioning, begins with the $k \times r$ generating array α and the intermediate array α^*. Now, however, the α-design is obtained by

replacing element j in the first row of α^* by the $(j+1)$th ordered element of S, and in the ith row by the $(j+1)$th element of C_i, where $j \in Z_s$.

For example, consider the design of the experiment with $v = 24$ treatments in blocks of size $k = 4$ in Example 2.1. The twenty-four treatment labels $0, 1, \ldots, 23$ are partitioned into the subgroup $S = \{0, 4, 8, 12, 16, 20\}$ and its three cosets $C_1 = \{1, 5, 9, 13, 17, 21\}$, $C_2 = \{2, 6, 10, 14, 18, 22\}$ and $C_3 = \{3, 7, 11, 15, 19, 23\}$. Hence, in the first row of α^* in Table 2.2 the elements $0, 1, \ldots, 5$ are replaced by $0, 4, \ldots, 20$ respectively. In the second row $0, 1, \ldots, 5$ are replaced by the elements of C_1, namely, $1, 5, \ldots, 21$ respectively. Continuing in this way gives the α-design in Table 2.4.

Table 2.4: α-Design obtained from α-array in Table 2.1 using alternative method

Replicate 1						Replicate 2						Replicate 3					
0	4	8	12	16	20	0	4	8	12	16	20	0	4	8	12	16	20
1	5	9	13	17	21	9	13	17	21	1	5	13	17	21	1	5	9
2	6	10	14	18	22	14	18	22	2	6	10	6	10	14	18	22	2
3	7	11	15	19	23	23	3	7	11	15	19	3	7	11	15	19	23

The α-design in Table 2.4 is isomorphic to that in Table 2.3. That is, by a simple relabelling, or permutation, of the treatment labels in Z_{24} one α-design can be obtained from the other.

It can be verified that if the rows of the treatment incidence matrix, N, correspond to the treatments in subset-coset order, i.e. the ordered treatments in S, C_1, C_2 and C_3 respectively, the treatment concurrence matrix for the α-design in Table 2.4 is identical to that in Figure 2.1. Consequently, the information and generalized inverse matrices are also the same for an α-design following this method of construction. If, however, the rows of the incidence matrix are lexicographically ordered the structure of the treatment concurrence matrix changes to

$$NN' = \sum_{u=0}^{s-1} \Gamma_h \otimes B_h$$

i.e. the order of the matrices Γ_h and B_h are reversed; see Jarrett and Hall (1978). Similarly, the information matrix changes to

$$A = \sum_{u=0}^{s-1} \Gamma_u^* \otimes A_u^*$$

and the generalized inverse of A to

$$A^{-} = \sum_{u=0}^{s-1} \Gamma_u^* \otimes A_u^{*-}$$

However, the definitions of the matrices B_h, A_u^* and A_u^{*-} do not change from those given in sections 2.3 and 2.5. These modifications have no effect on the average efficiency factor, E.

Chapter 3

α_n-Designs

3.1 Introduction

The family of α_n-designs belongs to the class of resolvable incomplete block designs. They are an extension of the family of α-designs in both their method of construction and structural properties. The $v = v_1 v_2 \ldots v_n$ treatments in an α_n-design are represented by n-tuples with the ith component of the n-tuple being at v_i levels $(i = 1, 2, \ldots, n)$. An α-design is a special case of an α_n-design with $n = 1$.

The family of α_n-designs is particularly well suited to the design of factorial experiments, and will be discussed in detail in the next chapter. They also enhance the class of resolvable block designs for a single treatment factor. More efficient designs can sometimes be obtained from α_n-designs with $n > 1$; see section 3.9.

The results in this chapter, and Chapters 4 and 5, are a novel extension of those presented in Chapter 2.

3.2 Design construction

The construction of an α_n-design is a straightforward extension of the method used to construct α-designs shown in section 2.2. However, in addition to the necessary constraint that the block size, k, divides the number of treatments, v, additional constraints must be satisfied for the construction of α_n-designs. This is that v and k can be factorised as $v = v_1 v_2 \ldots v_n$ and $k = k_1 k_2 \ldots k_n$ such that the ith factor

satisfies $v_i = k_i s_i$, where v_i, k_i and $s_i \in \mathbb{Z}^+$ $(i = 1, 2, \ldots, n)$.

For example, consider designs with $r = 3$ replicates of $v = 24$ treatments arranged in blocks of size $k = 4$. One possible factorisation of v and k satisfying $v_i = k_i s_i$ is $v_1 = 6$, $v_2 = 4$ and $k_1 = k_2 = 2$ so that $s_1 = 3$ and $s_2 = 2$ $(i = 1, 2)$. Since each parameter has been factorised into two components an α_2-design can be constructed.

As with α-designs, construction begins with a $k \times r$ generating array. For an α-design the candidate elements of this array are from the set Z_s. However, from the factorisation of design parameters given above, there is now a set Z_{s_1} associated with the first factor and Z_{s_2} with the second. Taking the direct sum over these sets yields a set of 2-tuples which is formed by concatenating the labels from each set, i.e. $Z_3 \oplus Z_2 = \{00, 01, 10, 11, 20, 21\}$. This set, which will be denoted $Z_{3,2}$, contains the candidate elements of the generating array of the α_2-design. Hence, a possible generating array α_2, where the subscript 2 denotes that the elements of the array are 2-tuples, is shown in Table 3.1.

Table 3.1: Generating array for $v_1 = 6$, $v_2 = 4$, $k_1 = k_2 = 2$ and $r = 3$

α_2-array		
00	00	00
11	20	21
01	11	21
20	21	00

A further five columns are generated from each column of the α_2-array by cyclic development of the components of each 2-tuple, beginning with the right-most component. The resulting 4×18 intermediate array α_2^* is shown in Table 3.2.

Let n-tuple addition be defined by

$$a_1 a_2 \ldots a_n + b_1 b_2 \ldots b_n = c_1 c_2 \ldots c_n \tag{3.1}$$

where $c_i = a_i + b_i$ modulo v_i $(i = 1, 2, \ldots, n)$. Now consider the set $< 3, 2 >= \{00, 02, 30, 32\}$ obtained by taking the direct sum over the subgroups $< 3 >= \{0, 3\}$, of Z_6, and $< 2 >= \{0, 2\}$, of Z_4, i.e. $< 3, 2 >=< 3 > \oplus < 2 >$. Addition of the ℓth

Table 3.2: Intermediate array generated from the α_2-array in Table 3.1

α_2^*

00	01	10	11	20	21	00	01	10	11	20	21	00	01	10	11	20	21
11	10	21	20	01	00	20	21	00	01	10	11	21	20	01	00	11	10
01	00	11	10	21	20	11	10	21	20	01	00	21	20	01	00	11	10
20	21	00	01	10	11	21	20	01	00	11	10	00	01	10	11	20	21

ordered element in the set $< 3, 2 >$ to each 2-tuple in the ℓth row of α_2^* yields the α_2-design shown in Table 3.3 ($\ell = 1, 2, 3, 4$).

Table 3.3: The α_2-design obtained from the α_2-array in Table 3.1

Replicate 1						Replicate 2						Replicate 3					
00	01	10	11	20	21	00	01	10	11	20	21	00	01	10	11	20	21
13	12	23	22	03	02	22	23	02	03	12	13	23	22	03	02	13	12
31	30	41	40	51	50	41	40	51	50	31	30	51	50	31	30	41	40
52	53	32	33	42	43	53	52	33	32	43	42	32	33	42	43	52	53

The columns of the α_2-design in Table 3.3 represent the blocks of the design. Notice that each block comprises one treatment from each of the sets $S_1 = \{00, 01, 10, 11, 20, 21\}$, $S_2 = \{02, 03, 12, 13, 22, 23\}$, $S_3 = \{30, 31, 40, 41, 50, 51\}$ and $S_4 = \{32, 33, 42, 43, 52, 53\}$, where the ℓth set is obtained by adding the ℓth ordered 2-tuple in $< 3, 2 >$ to each of the elements in $Z_{3,2}$. Further, these four sets are mutually disjoint and their union yields the set $Z_{6,4} = Z_6 \oplus Z_4$ consisting of twenty-four distinct 2-tuples, i.e. $S_1 \cup S_2 \cup S_3 \cup S_4 = Z_{6,4}$. Since each of the treatments in $Z_{6,4}$ is represented in each replicate, the α_2-design in Table 3.3 is resolvable.

In general, construction of an α_n-design begins with a $k \times r$ generating array α_n whose (ℓ, j)th element $\alpha_n(\ell, j)$ is contained in the set of n-tuples

$$Z_{s_1,s_2,\ldots,s_n} = \bigoplus_{i=1}^{n} Z_{s_i} = \{a_1 a_2 \ldots a_n : a_i \in Z_{s_i} \text{ for } i = 1, 2, \ldots, n\} \tag{3.2}$$

of order s, where $Z_{s_i} = \{0, 1, \ldots, s_i - 1\}$ ($\ell = 1, 2, \ldots, k$; $j = 1, 2, \ldots, r$). A further $s-1$ columns are generated from each column of the α_n-array by cyclic development of each component of the n-tuple in turn, starting with the right-most component a_n. This yields a $k \times rs$ intermediate array α_n^* whose elements are also in the set

$Z_{s_1,s_2,\ldots,s_n}$. Finally, the α_n-design is generated by adding the ℓth (lexicographically) ordered n-tuple in the set

$$< s_1, s_2, \ldots, s_n >= \bigoplus_{i=1}^{n} < s_i >= \{b_1 b_2 \ldots b_n : b_i \in < s_i > \text{ for } i = 1, 2, \ldots, n\} \quad (3.3)$$

of order k, where $< s_i >= \{0, s_i, \ldots, (k_i - 1)s_i\}$, to each n-tuple in the ℓth row of α_n^*.

The method of construction described above necessarily yields a resolvable design. For instance, let $b_1 b_2 \ldots b_n$ be the ℓth ordered n-tuple in $< s_1, s_2, \ldots, s_n >$, then its ith component, b_i, is a multiple of s_i, i.e. $b_i = d_i s_i$, where $d_i \in \{0, 1, \ldots, k_i - 1\}$ for $i = 1, 2, \ldots, n$. Now, let S_ℓ be the set of order s obtained by adding $b_1 b_2 \ldots b_n$ to every element in $Z_{s_1,s_2,\ldots,s_n}$. It follows that

$$S_\ell = \{c_1 c_2 \ldots c_n : c_i \in \{d_i s_i, d_i s_i + 1, \ldots, (d_i + 1)s_i - 1\}\} \quad (3.4)$$

If $\ell = 1$ then $b_1 b_2 \ldots b_n$ is the identity element $00 \ldots 0$ and hence, $S_1 = Z_{s_1,s_2,\ldots,s_n}$.

Now consider the set S_ℓ for $\ell > 1$. Without loss of generality, let the ℓth ordered element in $< s_1, s_2, \ldots, s_n >$ be such that its first $n - 1$ components are zero and its last component is $b_n = (\ell - 1)s_n$. Then,

$$S_\ell = \{c_1 c_2 \ldots c_n : c_i \in Z_{s_i} \text{ for } i = 1, 2, \ldots, n - 1 \text{ and } c_n \in Z_{s_n} + (\ell - 1)s_n\}$$

where

$$Z_{s_n} + (\ell - 1)s_n = \{(\ell - 1)s_n, (\ell - 1)s_n + 1, \ldots, \ell s_n - 1\}$$

Hence, although the first $n - 1$ components of the jth ordered n-tuple in S_ℓ are the same as the first $n - 1$ components of the jth ordered n-tuple in S_1, the nth component is distinct since $b_n \notin Z_{s_n}$ $(j = 1, 2, \ldots, s - 1)$. It follows that $S_1 \cap S_\ell$ is the empty set, i.e. S_1 and S_ℓ are mutually disjoint. Similarly, let $\ell' > 1$ and suppose $\ell \neq \ell'$. Then,

$$S_{\ell'} = \{c_1 c_2 \ldots c_n : c_i \in Z_{s_i} \text{ for } i = 1, 2, \ldots, n - 1 \text{ and } c_n \in Z_{s_n} + (\ell' - 1)s_n\}$$

where

$$Z_{s_n} + (\ell' - 1)s_n = \{(\ell' - 1)s_n, (\ell' - 1)s_n + 1, \ldots, \ell' s_n - 1\}$$

Since $\ell \neq \ell'$ it follows that the sets $Z_{s_n} + (\ell - 1)s_n$ and $Z_{s_n} + (\ell' - 1)s_n$ are mutually disjoint and hence, S_ℓ and S'_ℓ are also mutually disjoint.

From the method in which the intermediate array α_n^* is generated, it can be partitioned into r sets of k rows by s columns. Each row within a set comprises all of the elements in $Z_{s_1,s_2,\ldots,s_n}$. Now, since adding the ℓth ordered n-tuple in $< s_1, s_2, \ldots, s_n >$ to every element of $Z_{s_1,s_2,\ldots,s_n}$ yields k mutually disjoint subsets, it necessarily follows that adding the ℓth ordered n-tuple in $< s_1, s_2, \ldots, s_n >$ to every element in the kth row of α_n^* will yield a resolvable design.

3.3 α_n-designs isomorphic to α_m-designs ($m < n$)

Consider α_n-designs for the parametrisation $v = v_1 v_2 \ldots v_n$, $k = k_1 k_2 \ldots k_n$ and $s = s_1 s_2 \ldots s_n$, where $s_i = v_i/k_i$ for $i = 1, 2, \ldots, n$. It will be shown that if $s_i > 1$ for $i \leq m$, and $s_i = 1$ for $i > m$, an α_n-design is isomorphic to an α_m-design, where $m < n$.

For example, again consider designs with $v = 24$ treatments arranged in blocks of size $k = 4$. In section 3.2 an example was given in which v and k were factorised as $v_1 = 6$ and $v_2 = 4$ and $k_1 = k_2 = 2$. An alternative factorisation of the parameter k, satisfying $v_i = k_i s_i$, is $k_1 = 1$ and $k_2 = 4$ so that $s_1 = 6$ and $s_2 = 1$ $(i = 1, 2)$.

The candidate elements of the generating array α_2 are now contained in the set $Z_{6,1} = \{00, 10, 20, 30, 40, 50\}$. One possible generating array α_2 for this parametrisation is shown in Table 3.4.

Table 3.4: Generating array for $k_1 = 1$, $k_2 = 4$, $s_1 = 6$, $s_2 = 1$ and $r = 3$.

α_2-array		
30	10	30
20	30	10
50	50	00
10	00	50

The α_2-design obtained from the array in Table 3.4 is shown in Table 3.5.

Now, consider the rows within each replicate in Table 3.5. Across each row, the first digit is a cycle of length 6 of the integers (0 1 2 3 4 5). However, the second digit remains constant and is, therefore, redundant since it has no effect

Table 3.5: α_2-Design obtained from the α_2-array in Table 3.4

Replicate 1						Replicate 2						Replicate 3					
30	40	50	00	10	20	10	20	30	40	50	00	30	40	50	00	10	20
21	31	41	51	01	11	31	41	51	01	11	21	11	21	31	41	51	01
52	02	12	22	32	42	52	02	12	22	32	42	02	12	22	32	42	52
13	23	33	43	53	03	03	13	23	33	43	53	53	03	13	23	33	43

on the structure of the design. For example, let the 2-tuple $a_1a_2 \in Z_{6,4}$ and let $\rho : Z_{6,4} \to Z_{24}$ be the one-to-one function $\rho(a_1, a_2) = a_1 + a_2 s_1$ that maps each 2-tuple in $Z_{6,4}$ onto an integer in Z_{24}, i.e. each 2-tuple a_1a_2 in $Z_{6,4}$ is replaced by the integer $a_1 + a_2 s_1$ in Z_{24}. Applying the permutation ρ to the α_2-design in Table 3.5 yields the isomorphic α-design shown in Table 3.6. This design can be obtained directly from the generating array α_2 in Table 3.4 by dropping the second digit of each 2-tuple and using the parametrisation $v = 24$, $k = 4$ and $s = 6$.

Table 3.6: An α-design isomorphic to the α_2-design in Table 3.5

Replicate 1						Replicate 2						Replicate 3					
3	4	5	0	1	2	1	2	3	4	5	0	3	4	5	0	1	2
8	9	10	11	6	7	9	10	11	6	7	8	7	8	9	10	11	6
17	12	13	14	15	16	17	12	13	14	15	16	12	13	14	15	16	17
19	20	21	22	23	18	18	19	20	21	22	23	23	18	19	20	21	22

In contrast, each row within each replicate of the α_2-design in Table 3.3 has a nested cycling pattern. For example, the second digit in each pair of adjacent 2-tuples consists of a cycle of length $s_2 = 2$ of the integers (0 1) or (2 3) which is nested within a cycle of length $s_1 = 3$ of the integers (0 1 2) or (3 4 5) on the first digit. It is this nested cycling that gives an α_2-design its block-block circulant, or B^2C-structure. Note that there exists no permutation of the 2-tuples in this design that will yield an isomorphic α-design.

In general, consider α_n-designs for which the parametrisation of v, k and s is such that $s_i > 1$ for $i \leq m$ and $s_i = 1$ for $i > m$, and where $m < n$ $(i = 1, 2, \ldots, n)$. Now, let the n-tuple $a_1a_2 \ldots a_n \in Z_{v_1,v_2,\ldots,v_n}$ and let $\rho : Z_{v_m,v_{m+1},v_{m+2},\ldots,v_n} \to Z_\eta$ be

the one-to-one function

$$\rho(a_m, a_{m+1}, \ldots, a_n; s_m; v_{m+2}, v_{m+3}, \ldots, v_n) = a_m + s_m\Big(\sum_{i=m+1}^{n-1} a_i v(i) + a_n\Big)$$

that maps the last $n-m$ digits of each n-tuple in $Z_{v_1,v_2,\ldots,v_n}$ onto an integer $a_{m'} \in Z_\eta$, where $v(i) = v_{i+1}v_{i+2}\ldots v_n$ and $\eta = v_m v_{m+1} v_{m+2}\ldots v_n$, i.e. each n-tuple $a_1a_2\ldots a_n$ in $Z_{v_1,v_2,\ldots,v_n}$ is replaced by the m-tuple $a_1a_2\ldots a_{m-1}a_{m'}$. Applying the permutation ρ to the treatments in the α_n-design will yield an isomorphic α_m-design.

3.4 The treatment concurrence matrix

When the treatment incidence matrix, N, for an α-design is defined so that its rows correspond to the ordered treatments in Z_v, its treatment concurrence matrix NN' can be partitioned into k^2 $s \times s$ submatrices $M_{\ell m}$ whose (i,j)th element is the number of blocks in which the ith ordered element in S_ℓ and jth ordered element in S_m concur, where $S_\ell = \{(\ell-1)s, (\ell-1)s+1, \ldots, \ell s - 1\}$ $(i,j = 1,2,\ldots,s;\ \ell, m = 1,2,\ldots,k)$. Then, since the blocks within a replicate are obtained from the first block by cyclic substitution, modulo s, each row within a replicate consists of cyclic permutations of length s, or simply *cycles* of length s, of the integers $(d \quad d+1 \quad \ldots \quad d+s+1)$, where $d \in < s >$. Hence, the treatment concurrence matrix of an α-design is given by (2.5), i.e.

$$NN' = \sum_{h=0}^{s-1} B_h \otimes \Gamma_h \tag{3.5}$$

where B_h is defined in (2.5) and Γ_h is a basic circulant matrix of order s $(h = 0,1,\ldots,s-1)$. Note that although the B_h matrices in (3.5) contain the concurrences of each pair of treatments from subsets S_ℓ and S_m, the matrix Γ_h plays a key role in capturing the cyclical way in which pairs of treatments concur.

Now, consider an α_2-design and let the rows of N be ordered with respect to the subsets S_ℓ defined in (3.4) for $\ell = 1,2,\ldots,k$, with 2-tuples *lexicographically* ordered within subsets. It follows that the treatment concurrence matrix can again be partitioned into k^2 $s \times s$ submatrices $M_{\ell m}$, where now the (i,j)th element of any submatrix $M_{\ell m}$ is the number of blocks in which the ith and jth lexicographically

ordered 2-tuples in S_ℓ and S_m, respectively, concur, where S_ℓ and S_m are defined in (3.4) $(i, j = 1, 2, \ldots, s;\ \ell, m = 1, 2, \ldots, k)$.

For an α_2-design, blocks within a replicate are generated from the first block by cyclic substitution on each digit separately. This results in two levels of cycling. For instance, since $s = s_1 s_2$ the blocks within each replicate can be partitioned into s_1 sets of s_2 adjacent blocks. Each set of adjacent blocks consists of cycles of length s_2 of the integers $(d_2 \;\; d_2 + 1 \;\; \ldots \;\; d_2 + s_2 - 1)$, where $d_2 \in < s_2 >$. The first digit remains fixed within sets of adjacent blocks but consists of cycles of length s_1 of the integers $(d_1 \;\; d_1 + 1 \;\; \ldots \;\; d_1 + s_1 - 1)$ *across* sets, where $d_1 \in < s_1 >$. Hence, the method of construction results in cycles of length s_2 *nested* within cycles of length s_1. This nested cyclic structure manifests itself in the treatment concurrence matrix through the $s \times s$ submatrices $M_{\ell m}$ which can each be partitioned into s_1^2, $s_2 \times s_2$ submatrices. Hence, for an α_2-design the basic circulant matrix Γ_h in (3.5) is replaced by $\Gamma_{h_1} \otimes \Gamma_{h_2}$, where Γ_{h_i} is a basic circulant matrix of order s_i $(i = 1, 2)$. It follows that the concurrence matrix NN' of an α_2-design is given by

$$NN' = \sum_{h_1=0}^{s_1-1} \sum_{h_2=0}^{s_2-1} B_{h_1 h_2} \otimes \Gamma_{h_1} \otimes \Gamma_{h_2} \tag{3.6}$$

where $B_{h_1 h_2}$ is a $k \times k$ matrix.

For example, consider the α_2-design in Table 3.3 which has $s_1 = 3$ and $s_2 = 2$. The treatment concurrence matrix for this design is given in Figure 3.1. It is partitioned into sixteen 6×6 submatrices. The rows and columns of NN' are labelled in subset order, with treatments lexicographically ordered within subsets.

	00	01	10	11	20	21	02	03	12	13	22	23	30	31	40	41	50	51	32	33	42	43	52	53
00	3	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	1	1	0	0	0	1	1
01	0	3	0	0	0	0	0	0	1	0	1	1	1	0	1	0	1	0	0	1	0	0	1	1
10	0	0	3	0	0	0	1	1	0	0	0	1	0	1	0	1	0	1	1	1	1	0	0	0
11	0	0	0	3	0	0	1	1	0	0	1	0	1	0	1	0	1	0	1	1	0	1	0	0
20	0	0	0	0	3	0	0	1	1	1	0	0	0	1	0	1	0	1	0	0	1	1	1	0
21	0	0	0	0	0	3	1	0	1	1	0	0	1	0	1	0	1	0	0	0	1	1	0	1
−	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−
02	0	0	1	1	0	1	3	0	0	0	0	0	1	0	0	0	1	1	0	1	0	2	0	0
03	0	0	1	1	1	0	0	3	0	0	0	0	0	1	0	0	1	1	1	0	2	0	0	0
12	0	1	0	0	1	1	0	0	3	0	0	0	1	1	1	0	0	0	0	0	0	1	0	2
13	1	0	0	0	1	1	0	0	0	3	0	0	1	1	0	1	0	0	0	0	1	0	2	0
22	1	1	0	1	0	0	0	0	0	0	3	0	0	0	1	1	1	0	0	2	0	0	0	1
23	1	1	1	0	0	0	0	0	0	0	0	3	0	0	1	1	0	1	2	0	0	0	1	0
−	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−
30	0	1	0	1	0	1	1	0	1	1	0	0	3	0	0	0	0	0	0	0	1	1	0	1
31	1	0	1	0	1	0	0	1	1	1	0	0	0	3	0	0	0	0	0	0	1	1	1	0
40	0	1	0	1	0	1	0	0	1	0	1	1	0	0	3	0	0	0	0	1	0	0	1	1
41	1	0	1	0	1	0	0	0	0	1	1	1	0	0	0	3	0	0	1	0	0	0	1	1
50	0	1	0	1	0	1	1	1	0	0	1	0	0	0	0	0	3	0	1	1	0	1	0	0
51	1	0	1	0	1	0	1	1	0	0	0	1	0	0	0	0	0	3	1	1	1	0	0	0
−	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	+	−	−	−	−	−	−	−
32	1	0	1	1	0	0	0	1	0	0	0	2	0	0	0	1	1	1	3	0	0	0	0	0
33	0	1	1	1	0	0	1	0	0	0	2	0	0	0	1	0	1	1	0	3	0	0	0	0
42	0	0	1	0	1	1	0	2	0	1	0	0	1	1	0	0	0	1	0	0	3	0	0	0
43	0	0	0	1	1	1	2	0	1	0	0	0	1	1	0	0	1	0	0	0	0	3	0	0
52	1	1	0	0	1	0	0	0	0	2	0	1	0	1	1	1	0	0	0	0	0	0	3	0
53	1	1	0	0	0	1	0	0	2	0	1	0	1	0	1	1	0	0	0	0	0	0	0	3

Figure 3.1: Treatment concurrence matrix for the α_2-design in Table 3.3.

This matrix is given by (3.6) where

$$B_{00} = \begin{pmatrix} 3 & 0 & 0 & 1 \\ 0 & 3 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 1 & 0 & 0 & 3 \end{pmatrix} \quad B_{01} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$B_{10} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad B_{11} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$B_{20} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad B_{21} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \end{pmatrix}$$

For this design, for instance, the first term in the expanded series in (3.6) is $B_{00} \otimes \Gamma_0 \otimes \Gamma_0$ and gives the diagonal elements of each 6×6 submatrix in Figure 3.1. If each submatrix in Figure 3.1 is partitioned into 3^2, 2×2 submatrices then it is easy to see that $B_{01} \otimes \Gamma_0 \otimes \Gamma_1$, for example, gives the off-diagonal elements of each 2×2 matrix along the leading diagonal of each submatrix $M_{\ell m}$ for $\ell, m = 1, 2, 3, 4$. Further, consider the submatrix M_{24} in Figure 3.1, i.e.

$$M_{24} = \begin{array}{c} \\ 02 \\ 03 \\ - \\ 12 \\ 13 \\ - \\ 22 \\ 23 \end{array} \begin{array}{c} \begin{array}{cc|cc|cc} 32 & 33 & 42 & 43 & 52 & 53 \end{array} \\ \left(\begin{array}{cc|cc|cc} 0 & 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 \\ - & - & - & - & - & - \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ - & - & - & - & - & - \\ 0 & 2 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \end{array}\right) \end{array}$$

It is obtained by pre-multiplying the matrix $\Gamma_{h_1} \otimes \Gamma_{h_2}$ by the treatment concurrence in the second row and fourth column of each $B_{h_1 h_2}$ matrix for appropriate values of h_1 and h_2 ($h_i = 0, 1, \ldots, s_i - 1; i = 1, 2$). For instance, element (2,4) in B_{11} is 2. Hence, the (1,2)th, (2,3)th and (3,1)th 2×2 submatrices in M_{24} each have a 2 in the off-diagonal positions and are obtained by pre-multiplying $\Gamma_1 \otimes \Gamma_1$ by 2.

More generally, let the rows of the treatment incidence matrix of an α_n-design correspond with the lexicographically ordered treatments in $S_1, S_2, \ldots, S_k$ respectively, where S_i is defined in (3.4). Then, for an α_n-design with $s = s_1 s_2 \ldots s_n$ the basic circulant matrix Γ_h in (3.5) is replaced by

$$\Gamma_{h_1} \otimes \Gamma_{h_2} \otimes \cdots \otimes \Gamma_{h_n} \tag{3.7}$$

where Γ_{h_i} is a basic circulant matrix of order s_i ($i = 1, 2, \ldots, n$). Hence, the treatment concurrence matrix of an α_n-design is given by

$$NN' = \sum_{h_1=0}^{s_1-1} \sum_{h_2=0}^{s_2-1} \cdots \sum_{h_n=0}^{s_n-1} B_{h_1 h_2 \ldots h_n} \otimes \Gamma_{h_1} \otimes \Gamma_{h_2} \otimes \cdots \otimes \Gamma_{h_n} \tag{3.8}$$

where $B_{h_1 h_2 \ldots h_n}$ is a $k \times k$ matrix whose (ℓ, m)th element is the number of concurrences of all pairs of treatments $a_1 a_2 \ldots a_n \in S_\ell$ and $b_1 b_2 \ldots b_n \in S_m$ ($\ell, m = 1, 2, \ldots, k$). The treatment concurrence matrix in (3.8) has $B^n C$-structure.

3.5 Generating NN' from the α_n-array

Like α-designs, the treatment concurrence matrix of an α_n-design can be obtained directly from its generating array α_n. For instance, consider an α_n-design with $v = v_1v_2\ldots v_n$ treatments arranged in blocks of size $k = k_1k_2\ldots k_n$ and $s = s_1s_2\ldots s_n$ blocks per replicate. Now, suppose the n-tuples $a_1a_2\ldots a_n \in S_\ell$ and $b_1b_2\ldots b_n \in S_m$ concur in the same block of the design $(\ell, m = 1, 2, \ldots, k)$. Further, suppose that the n-tuples $p_1p_2\ldots p_n$ and $q_1q_2\ldots q_n$ occur in the ℓth and mth rows, respectively, of the corresponding column of the intermediate array α_n^* from which the design was generated, i.e. $p_1p_2\ldots p_n, q_1q_2\ldots q_n \in Z_{s_1,s_2,\ldots,s_n}$. From the method of construction it follows that $a_1a_2\ldots a_n = c_1c_2\ldots c_n + p_1p_2\ldots p_n$ and $b_1b_2\ldots b_n = d_1d_2\ldots d_n + q_1q_2\ldots q_n$, where $c_1c_2\ldots c_n, d_1d_2\ldots d_n \in < s_1, s_2, \ldots, s_n >$. Hence, $c_i = (\ell - 1)s_i$ and $d_i = (m-1)s_i$ for $i = 1, 2$.

Since c_i and d_i are multiples of s_i, and from (2.1), it follows that n-tuple subtraction is defined by

$$b_1b_2\ldots b_n - a_1a_2\ldots a_n = h_1h_2\ldots h_n \tag{3.9}$$

where

$$h_i = q_i - p_i \text{ modulo } s_i = \begin{cases} q_i - p_i & q_i \geq p_i \\ s_i - |q_i - p_i| & q_i < p_i \end{cases} \tag{3.10}$$

for $h_i, p_i, q_i \in Z_{s_i}$. Hence, from (3.9) and (3.10) it follows that

$$b_1b_2\ldots b_n - a_1a_2\ldots a_n = q_1q_2\ldots q_n - p_1p_2\ldots p_n = h_1h_2\ldots h_n$$

Since each set of s adjacent columns in α_n^* is generated from the first by cyclic development of each digit in turn, it follows that the difference between pairs of 2-tuples in the ℓth and mth rows of each successive column in that set also equals $h_1h_2\ldots h_n$. Furthermore, since the first column of the jth set in α_n^* is the jth column of the generating array α_n for $j = 1, 2, \ldots, r$, it follows that these differences can be obtained directly from α_n.

For example, consider the α_2-design in Table 3.3. Treatments 22 and 41 in the second and third rows, respectively, of the first block of replicate 2 can be factorised

as $22 = 02 + p_1p_2$ and $41 = 30 + q_1q_2$, where $p_1p_2 = 20$ and $q_1q_2 = 11$. From (3.9) and (3.10) it follows that $41 - 22 = 11 - 20 = 21$. Furthermore, the difference between each successive pair of treatments in the remaining five blocks of replicate 2, or equivalently each successive pair of 2-tuples in the corresponding rows and columns of the intermediate array α_2^* in Table 3.2, is also 21. Consequently, all pairs of treatments $a_1a_2 \in S_\ell$ and $b_1b_2 \in S_m$ whose difference is 21 concur within a block of replicate 2. Hence, this information could have been obtained directly from the second column of the generating array α_2 in Table 3.1.

3.6 The information and generalized inverse matrices

Since α_n-designs are resolvable incomplete block designs with constant block size k and each treatment replicated r times, the intra-block information matrix is again given by

$$A = rI_v - NN'/k$$

Re-expressing A in terms of the treatment concurrence matrix defined in (3.8) gives

$$A = \sum_{h_1=0}^{s_1-1} \sum_{h_2=0}^{s_2-1} \cdots \sum_{h_n=0}^{s_n-1} A_{h_1h_2\ldots h_n} \otimes \Gamma_{h_1} \otimes \Gamma_{h_2} \otimes \cdots \otimes \Gamma_{h_n} \tag{3.11}$$

where

$$A_{h_1h_2\ldots h_n} = \begin{cases} rI_k - B_{h_1h_2\ldots h_n}/k & \text{if } h_i = 0,\ \forall i \in \{1, 2, \ldots, n\} \\ -B_{h_1h_2\ldots h_n}/k & \text{otherwise} \end{cases} \tag{3.12}$$

and where $B_{h_1h_2\ldots h_n}$ is defined in (3.8).

From (2.8) each $s_j \times s_j$ basic circulant matrix Γ_{h_j} can be written as

$$\Gamma_{h_j} = \sum_{u_j=0}^{s_j-1} \omega_{u_j}^{h_j} \Gamma_{u_j}^* \tag{3.13}$$

where $\omega_{u_j} = \exp(2\pi i u_j/s_j)$ is the u_jth of the s_jth roots of unity. Substituting the expression for the jth basic circulant matrix given in (3.13) into (3.11) gives

$$A = \sum_{u_1=0}^{s_1-1} \sum_{u_2=0}^{s_2-1} \cdots \sum_{u_n=0}^{s_n-1} A_{u_1u_2\ldots u_n}^* \otimes \Gamma_{u_1}^* \otimes \Gamma_{u_2}^* \otimes \cdots \otimes \Gamma_{u_n}^* \tag{3.14}$$

where

$$A^*_{u_1u_2\ldots u_n} = \sum_{h_1=0}^{s_1-1}\sum_{h_2=0}^{s_2-1}\cdots\sum_{h_n=0}^{s_n-1}\omega_{u_1}^{h_1}\omega_{u_2}^{h_2}\ldots\omega_{u_n}^{h_n}A_{h_1h_2\ldots h_n} \tag{3.15}$$

and where the mutually orthogonal idempotent matrices $\Gamma^*_{u_j}$ are defined in (2.9). From (3.14) the information matrix of an α_n-design can be seen to be a straightforward extension of the information matrix of an α-design given in (2.11).

The information matrix of an α_n-design has the same properties as that of an α-design; see subsection 2.5.1. For instance, from (3.15) and (3.12), and since $\sum_j\sum_{h_j} B_{h_1h_2\ldots h_n} = rJ_k$, then $A^*_{00\ldots0} = r(I_k - K_k)$. Hence, $\text{rank}(A^*_{00\ldots0}) = k-1$ so that $A^{*-}_{00\ldots0} = I_k/r$. It again follows for a connected design that the remaining $s-1$, $k\times k$ $A^*_{u_1u_2\ldots u_n}$ matrices for which $u_1+u_2+\cdots+u_n \neq 0$ are each of full rank so that $A^{*-}_{u_1u_2\ldots u_n} = A^{*-1}_{u_1u_2\ldots u_n}$. Furthermore, the matrices $A^{*-1}_{u_1u_2\ldots u_n}$ are Hermitian; see Theorem B.10 and its proof in Appendix B.

Since $\Gamma^*_{u_j}$ and $\Gamma^*_{u_{j'}}$ $(j \neq j')$ are mutually orthogonal and idempotent, it follows from Theorem 2.1 that

$$A^- = \sum_{u_1=0}^{s_1-1}\sum_{u_2=0}^{s_2-1}\cdots\sum_{u_n=0}^{s_n-1}A^{*-}_{u_1u_2\ldots u_n}\otimes\Gamma^*_{u_1}\otimes\Gamma^*_{u_2}\otimes\cdots\otimes\Gamma^*_{u_n} \tag{3.16}$$

is a generalized inverse of A, where

$$A^{*-}_{u_1u_2\ldots u_n} = \begin{cases} I_k/r & \text{if } u_j = 0,\ \forall j\in\{1,2,\ldots,n\} \\ A^{*-1}_{u_1u_2\ldots u_n} & \text{otherwise} \end{cases} \tag{3.17}$$

3.7 The average efficiency factor, E

For the contrast matrix $C = I_v - K_v$, the average efficiency factor is given by (1.7), namely

$$E = \frac{v-1}{r\text{trace}(CA^-)}$$

Let C be written as

$$(I_{k_1}\otimes\cdots\otimes I_{k_n})\otimes(I_{s_1}\otimes\cdots\otimes I_{s_n}) - (K_{k_1}\otimes\cdots\otimes K_{k_n})\otimes(K_{s_1}\otimes\cdots\otimes K_{s_n})$$

Then, the matrix product CA^- equals

$$\sum_{u_1=0}^{s_1-1}\sum_{u_2=0}^{s_2-1}\cdots\sum_{u_n=0}^{s_n-1}\left[A^{*-}_{u_1u_2\ldots u_n}\otimes\left(\bigotimes_{j=1}^{n}I_{s_i}\Gamma^*_{u_i}\right) - K_kA^{*-}_{u_1u_2\ldots u_n}\otimes\left(\bigotimes_{j=1}^{n}K_{s_i}\Gamma^*_{u_i}\right)\right] \tag{3.18}$$

Extending the results given in (2.17) so that

$$L_{s_j}\Gamma^*_{u_j} = \begin{cases} K_{s_j} & \text{if } u_j = 0 \text{ and } L_{s_j} = I_{s_j} \text{ or } K_{s_j} \\ \Gamma^*_{u_j} & \text{if } u_j > 0 \text{ and } L_{s_j} = I_{s_j} \\ 0_{s_j} & \text{if } u_j > 0 \text{ and } L_{s_j} = K_{s_j} \end{cases} \tag{3.19}$$

the expression for CA^- given in (3.18) can now be simplified. The proof of these results is analogous to the proof following (2.17) and is therefore omitted here.

It now follows from (3.19) that CA^- can be partitioned into two parts: the contribution when $u_j = 0$ for all j and the contribution when at least one $u_j > 0$. From (3.17), $A^{*-}_{00\ldots0} = I_k/r$ so that the contribution to CA^- is

$$\frac{1}{r}(I_k - K_k) \otimes K_s$$

When at least one u_j is non-zero, the term on the right hand side of the minus sign in (3.18) is zero and so does not contribute to CA^-. Thus, each term for which $u_1 + u_2 + \cdots + u_n \neq 0$ makes the contribution

$$A^{*-1}_{u_1u_2\ldots u_n} \otimes \left(\Gamma^*_{u_1} \otimes \Gamma^*_{u_2} \otimes \cdots \otimes \Gamma^*_{u_n}\right)$$

to CA^-. Since $\text{trace}(\Gamma^*_{u_j}) = 1$, then

$$\text{trace}(CA^-) = \frac{1}{r}(k-1) + \sum_u \text{trace}\left(A^{*-1}_{u_1u_2\ldots u_n}\right) \tag{3.20}$$

where $\sum_u$ is the summation over all combinations of $u_1u_2\ldots u_n$ excluding $u_1 = u_2 = \cdots = u_n = 0$. Substituting (3.20) in (2.15) gives

$$E = \frac{v-1}{(k-1) + r\sum_u \text{trace}\left(A^{*-1}_{u_1u_2\ldots u_n}\right)}$$

Thus, the average efficiency factor for an α_n-design is analogous to that for an α-design.

Again, complex matrix inversion can be avoided in the calculation of E. This is because $A^*_{u_1u_2\ldots u_n}$ and its inverse are Hermitian matrices. By a straightforward extension of the results at the end of section 2.6, it follows that

$$E = \frac{v-1}{(k-1) + r\sum_u \text{trace}\left[(X_{1u_1u_2\ldots u_n} + X_{2u_1u_2\ldots u_n}X^{-1}_{1u_1u_2\ldots u_n}X_{2u_1u_2\ldots u_n})^{-1}\right]}$$

where the $k \times k$ matrices $X_{1u_1u_2\ldots u_n}$ and $X_{2u_1u_2\ldots u_n}$ are the real and imaginary parts, respectively, of $A^*_{u_1u_2\ldots u_n}$. Hence, E requires similar calculations to those required for an α-design.

3.8 A counting rule

In section 2.7 a counting rule was derived for the screening of efficient α-designs. This rule, based on the (M, S)-optimality criterion, used the inherent BC-structure in the treatment concurrence matrix of α-designs to obtain substantial reductions in the number of multiplications required to calculate trace$[(NN')^2]$. The B^nC-structure in the treatment concurrence matrix of α_n-designs will now be used to derive a similar screening function.

As with α-designs, the first step of the (M, S)-optimality criterion does not discriminate between designs of the same size since trace(A) equals the constant $v(k-1)/k$. Therefore, only the second stage of the process which involves screening for designs with minimum trace(A^2) is carried out. In section 2.7 this was shown to be equivalent to minimising the sum of squared off-diagonal elements of NN'. For a resolvable incomplete block design this is given by

$$\text{trace}[(A/r)^2] = v(k-1)^2/k^2 + \sum_{i \neq j} \beta_{ij}^2/r^2k^2$$

where β_{ij} is the number of blocks in which treatments i and j concur.

For an α_n-design

$$\begin{aligned} \text{trace}\left[(NN')^2\right] &= \text{trace}\left[\left(\sum_{h_1=0}^{s_1-1} \cdots \sum_{h_n=0}^{s_n-1} B_{h_1 \ldots h_n} \otimes \Gamma_{h_1} \otimes \cdots \otimes \Gamma_{h_n}\right)^2\right] \\ &= \sum_{g_1=0}^{s_1-1} \cdots \sum_{g_n=0}^{s_n-1} \sum_{h_1=0}^{s_1-1} \cdots \sum_{h_n=0}^{s_n-1} \text{trace}\left(B_{g_1 \ldots g_n} B_{h_1 \ldots h_n}\right) \prod_{j=1}^{n} \text{trace}\left(\Gamma_{g_j} \Gamma_{h_j}\right) \end{aligned}$$

where the $k \times k$ matrices $B_{g_1 \ldots g_n}$ and $B_{h_1 \ldots h_n}$ are defined as in (3.8). Since

$$\text{trace}\left(\Gamma_{g_j} \Gamma_{h_j}\right) = \begin{cases} s_j & \text{if } h_j = s_j - g_j \\ 0 & \text{otherwise} \end{cases}$$

then

$$\prod_{j=1}^{n} \text{trace}\left(\Gamma_{g_j} \Gamma_{h_j}\right) = \begin{cases} s & \text{if } h_j = s_j - g_j \ \forall j \in \{1, \ldots, n\} \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, since $B_{h_1 \ldots h_n} = B'_{g_1 \ldots g_n}$ when $h_j = s_j - g_j$, it follows that

$$\text{trace}\left[(NN')^2\right] = s \sum_{g_1=0}^{s_1-1} \cdots \sum_{g_n=0}^{s_n-1} \sum_{\forall \ell, m} \beta_{\ell,m}^2(g_1 \ldots g_n)$$

where $\sum_{\forall \ell,m}$ is the summation over all combinations of ℓ and m and $\beta_{\ell m}(g_1 \ldots g_n)$ is the (ℓ, m)th element of $B_{g_1 \ldots g_n}$ $(\ell, m = 1, 2, \ldots, k;\ g_i = 0, 1, \ldots, s_i - 1;\ i = 1, 2, \ldots, n)$. Hence, the sum of squared off-diagonal elements of the treatment concurrence matrix of an α_n-design is given by

$$\sum_{i \neq j} \lambda_{ij}^2 = s \left[\sum_{\ell \neq m} \beta_{\ell m}^2(0 \ldots 0) + \sum_{g_1=0}^{s_1-1} \cdots \sum_{g_n=0}^{s_n-1} \sum_{\ell \neq m} \beta_{\ell m}^2(g_1 \ldots g_n) \right]$$

where $\sum_{\ell \neq m}$ is the summation over all combinations of ℓ and m except $\ell = m = 1$. Hence, the sum of squared off-diagonal elements of NN' is given by s times the sum of squared off-diagonal elements of each $k \times k$ matrix $B_{g_1 \ldots g_n}$ $(g_i = 0, 1, \ldots, s_i - 1;\ i = 1, 2, \ldots, n)$.

It is possible that further computational savings can be made by exploiting the symmetry of the $B_{g_1 \ldots g_n}$ matrices; that is, by using the property $B_{h_1 \ldots h_n} = B'_{g_1 \ldots g_n}$ for $h_i = s_i - g_i$. However, this does not appear as trivial as in the case of an α-design and is a problem for future investigation.

3.9 Enhancing α-designs

Efficient search procedures operating on the generating array together with the simple calculation of E to assess each new design makes the class of α-designs particularly useful in practice. For $n > 1$, α_n-designs can be used to supplement α-designs since they can also be obtained from a generating array and the calculation of E is simple and quick. Furthermore, by using the wider range of α_n-designs, the best available α-designs can sometimes be improved upon.

For example, the α_2-design for $v = 12$ and $r = k = 3$ constructed from the generating array in Table 3.7, with $s_1 = s_2 = 2$ and $k_1 = 3$, $k_2 = 1$, is a triple rectangular lattice design and hence, is known to be optimal ($E = 0.6801$). The best α-design of this size had average efficiency factor $E = 0.6720$. It was found after extensive runs of CycDesigN (Whitaker, Williams and John, 1997) which is a computer package for the generation of optimal or near-optimal experimental designs. There are many other instances where square lattice and rectangular lattice designs cannot be written as α-designs but exist as α_2-designs, e.g. $r = 3$, $s = 4, 8$ and $r = 4$, $s = 4, 8, 9$.

Table 3.7: Generating array for a triple rectangular lattice α_2-design

01	11	10
10	01	11
11	10	01

Even in the more common cases where α-designs need to be constructed for parameter sets where the optimal design has not been ascertained analytically, α_n-designs ($n > 1$) provide worthwhile alternatives over the best α-designs. For instance, for $v = 32$, $k = 8$ and $r = 6$ the best α-design has $E = 0.8947$ compared with $E = 0.8960$ for the best α_2-design which in fact attains a known upper bound for these parameters.

Table 3.8: Optimal α_2-designs of size $(v, k = k_1 k_2, r)$

v	k	k_1	k_2	r	E_α	E_{α_2}
16	4	2	2	2	0.7143	0.7143
16	4	2	2	3	0.7538	0.7692
28	7	7	1	5	0.8747	0.8756
28	7	7	1	6	0.8790	0.8801
32	8	8	1	5	0.8911	0.8921
32	8	4	2	6	0.8947	0.8960
32	8	4	2	7	0.8973	0.8986

Further examples of cases in which the average efficiency factor, E_{α_2}, of the best available α_2-design of size v, k and r exceeds the average efficiency factor, E_α, of the best α-design of the same size are given in Table 3.8. In each case, not only does E_{α_2} exceed E_α, it also attains a known upper bound. Extensive runs of CycDesigN were used to find the best α-design for each set of design parameters considered in Table 3.8. Competing α_2-designs of the same size were generated by a computer algorithm based on a program used in CycDesigN for the generation of efficient α-designs. This program was extensively modified to generate and assess competing α_n-designs. A copy of the C/C++ code for this program can be found in Appendix D.

3.10 Alternative construction method

The alternative method for constructing α-designs presented in section 2.8 will now be extended to α_n-designs. This method is useful in demonstrating the relationship that exists between n-cyclic designs and α_n-designs; see section 5.3.

Consider again designs for which the parameters v, k and s can be factorised as $v = v_1 v_2 \dots v_n$, $k = k_1 k_2 \dots k_n$ and $s = s_1 s_2 \dots s_n$, where $v_i = k_i s_i$ $(i = 1, 2, \dots, n)$. Associated with the ith factor in this parametrisation is the group Z_{v_i}, closed under addition modulo v_i, and its subgroup $< k_i >= \{0, k_i, \dots, (s_i - 1)k_i\}$. The v treatments in the design are labelled with the n-tuples in the set $Z_{v_1, v_2, \dots, v_n}$.

The set $Z_{v_1, v_2, \dots, v_n}$ can be partitioned into k mutually disjoint subsets of order s by a straightforward extension of the method described in section 2.8. The first subset is given by

$$S = \{a_1 a_2 \dots a_n : a_i \in < k_i > \text{ for } i = 1, 2, \dots, n\}$$

It is a subgroup of $Z_{v_1, v_2, \dots, v_n}$ under the n-tuple addition defined in (3.1) and is obtained by taking the direct sum over all subgroups $< k_i >$. The $k - 1$ remaining subsets C_j are obtained by adding the jth ordered element in the set $Z_{k_1, k_2, \dots, k_n}$ to every element in S, i.e.

$$C_j = \{b_1 b_2 \dots b_n : b_i \in < k_i > + d_i \text{ and } d_i \in Z_{k_i} \text{ for } i = 1, 2, \dots, n\}$$

where $< k_i > + d_i = \{d_i, k_i + d_i, \dots, (s_i - 1)k_i + d_i\}$. The subsets C_j are cosets of S $(j = 1, 2, \dots, k - 1)$.

Construction begins with a $k \times r$ generating array α_n and the intermediate array α_n^* whose elements are n-tuples from the set $Z_{s_1, s_2, \dots, s_n} = \{a_1 a_2 \dots a_n : a_i \in Z_{s_i}\}$. The α_n-design is obtained by substituting the n-tuples $a_1 a_2 \dots a_n$ in the first row of α_n^* with the pth (lexicographically) ordered element in S, and the n-tuples in the jth row with the pth ordered element in C_j $(j = 1, 2, \dots, k - 1;\ p = 1, 2, \dots, s)$.

Example 3.1 *Consider again experiments arranged in $r = 3$ replicates of $v = 24$ in blocks of size $k = 4$. Let the parameters v and k be factorised as $v_1 = 6$, $v_2 = 4$ and $k_1 = k_2 = 2$, so that $s_1 = 3$ and $s_2 = 2$. Then the generating array α_2 and*

intermediate array α_2^* in Tables 3.1 and 3.2, respectively, can be used to generate an α_2-design following the alternative method of construction described above. The intermediate array is reproduced in Table 3.9 for convenience. Note that the elements of α_2^* are from the set $Z_{3,2} = \{00, 01, 10, 11, 20, 21\}$, as required.

Table 3.9: Intermediate array generated from the α_2-array in Table 3.1

α_2^*																	
00	01	10	11	20	21	00	01	10	11	20	21	00	01	10	11	20	21
11	10	21	20	01	00	20	21	00	01	10	11	21	20	01	00	11	10
01	00	11	10	21	20	11	10	21	20	01	00	21	20	01	00	11	10
20	21	00	01	10	11	21	20	01	00	11	10	00	01	10	11	20	21

From the above parametrisation, the group Z_6 and its subgroup $< 2 >= \{0, 2, 4\}$ are associated with the first factor and the group Z_4 and its subgroup $< 2 >= \{0, 2\}$ with the second. The twenty-four treatment labels are contained in the set

$$Z_{6,4} = \{00,01,02,03,10,11,12,13,20,21,22,23,30,31,32,33,40,41,42,43,50,51,52,53\}$$

Hence, a subgroup of $Z_{6,4}$ is given by the set $S = \{00, 02, 20, 22, 40, 42\}$. The three cosets of S are $C_1 = \{01, 03, 21, 23, 41, 43\}$, $C_2 = \{10, 12, 30, 32, 50, 52\}$ and $C_3 = \{11, 13, 31, 33, 51, 53\}$.

Now, let $a_1a_2 \in Z_{3,2}$ and let $p = a_1 + a_2s_1$. Then, the α_2-design shown in Table 3.10 is obtained by substituting the 2-tuples 00, 01, 10, 11, 20 and 21 with 00, 02, 20, 22, 40 and 42 respectively in the first row of α_2^*, 01, 03, 21, 23, 41 and 43 respectively in the second row, 10, 12, 30, 32, 50 and 52 respectively in the third and 11, 13, 31, 33, 51 and 53 respectively in the last. This labelling system is a simple permutation of that used in section 3.2 to obtain the α_2-design in Table 3.3. Hence, these two designs are isomorphic.

Note that each replicate in Table 3.10 can be generated from its initial block by adding each 2-tuple in the subgroup S in turn to every 2-tuple in the initial block.

Once again, if the rows of the treatment incidence matrix are arranged with respect to the treatments in subset-coset order then the treatment concurrence matrix, and hence the information and generalized inverse matrices, for an α_n-design

Table 3.10: α_2-Design for $v_1 = 6$, $v_2 = 4$, $k_1 = k_1 = 2$, and $r = 3$ obtained by alternative method

Replicate 1						Replicate 2						Replicate 3					
00	02	20	22	40	42	00	02	20	22	40	42	00	02	20	22	40	42
23	21	43	41	17	21	41	43	01	03	21	23	43	41	03	01	23	21
12	10	32	30	52	50	32	30	52	50	12	10	52	50	12	10	32	30
51	53	11	13	31	33	53	51	13	11	33	31	51	53	11	13	31	33

constructed in the manner described in this section is identical to that of an α_n-design constructed following the method described in section 3.2. If, however, the rows of the incidence matrix are ordered by the first ordered treatment in the sets $S, C_1, \ldots, C_{k-1}$, followed by the second ordered treatment in each set, and so on, the structure of the treatment concurrence matrix is given by

$$NN' = \sum_{h_1=0}^{s_1-1} \sum_{h_2=0}^{s_2-1} \cdots \sum_{h_n=0}^{s_n-1} \Gamma_{h_1} \otimes \Gamma_{h_2} \otimes \cdots \otimes \Gamma_{h_n} \otimes B_{h_1 h_2 \ldots h_n}$$

where $B_{h_1 h_2 \ldots h_n}$ is the same $k \times k$ matrix defined in (3.8). Note that, again, the $B_{h_1 h_2 \ldots h_n}$ matrix is the last term in the Kronecker product. This modification filters through to the information and generalized inverse matrices but does not impact on the average efficiency factor, E.

Chapter 4

α_n-Designs and factorial experiments

4.1 Introduction

In the previous chapter a general methodology for the construction of α_n-designs was developed. It was also shown that the treatment concurrence matrix and hence, information matrix of α_n-designs have B^nC-structure; see sections 3.4 and 3.6. This property plays an important role in this chapter in establishing criteria for the generation of efficient α_n-designs for factorial experiments.

Section 4.2 introduces some of the terminology and notation that is used throughout the chapter. Then, in section 4.3, a set of contrast matrices are defined which partition each factorial treatment combination into components representing main effects and interactions. The structure of these matrices conforms with the B^nC-structure of the information matrix. This enables, in section 4.4, the derivation of a closed form mathematical expression for the average efficiency factor of any generalized interaction for an α_n-design. This expression is shown to yield substantial savings in the calculations required to compute these average efficiency factors. Section 4.5 shows how these efficiency factors can be obtained recursively so that further reductions in computational effort are achieved. Finally, in section 4.6 it is shown that α_n-designs do not, in general, possess orthogonal factorial structure.

4.2 Preliminaries

Consider a factorial experiment with n factors $F_1, F_2, \ldots, F_n$ where the ith factor has v_i levels and $v = v_1 v_2 \ldots v_n$. The v factorial treatment combinations are written as n-tuples, $a_1 a_2 \ldots a_n$ where $a_i \in Z_{v_i}$ $(i = 1, 2, \ldots, n)$. Let any generalized interaction be represented by $F^x = F_1^{x_1} F_2^{x_2} \ldots F_n^{x_n}$, where $x = x_1 x_2 \ldots x_n$ and $x_i = 1$ if factor F_i is present in the interaction and $x_i = 0$ otherwise.

Now, let the factorial experiment be arranged in a resolvable block design with r replicates each with s blocks of size k, such that each treatment combination occurs once in each replicate. Then, the intrablock model is given by

$$y_{i_1 i_2 \ldots i_n pj} = \mu + \rho_p + \beta_{pj} + \tau_{i_1 i_2 \ldots i_n} + \epsilon_{i_1 i_2 \ldots i_n pj} \tag{4.1}$$

$$(i_m = 0, 1, \ldots, v_m - 1; m = 1, 2, \ldots, n; p = 1, 2 \ldots, r; j = 1, 2, \ldots, s)$$

where $y_{i_1 i_2 \ldots i_n pj}$ is the observed response when the treatment combination $i_1 i_2 \ldots i_n$ is applied to the jth block of the pth replicate, μ is the overall mean effect, $\tau_{i_1 i_2 \ldots i_n}$ is the effect of the treatment combination $i_1 i_2 \ldots i_n$, ρ_p is the effect of the pth replicate, β_{pj} is the effect of the jth block in the pth replicate and $\epsilon_{i_1 i_2 \ldots i_n pj}$ are independent and normally distributed random variables with means zero and variances σ^2.

Let C_x be a $v \times v$ contrast matrix that spans the factorial space of the generalized interaction F^x and let τ be a $v \times 1$ vector of treatment parameters $\tau_{i_1 i_2 \ldots i_n}$ $(i_m = 0, 1, \ldots, v_m - 1; m = 1, 2, \ldots n)$. Then, $C_x \hat{\tau}$ is an unbiased estimator of the estimable function $C_x \tau$, where $\hat{\tau}$ is defined as in (1.2). The variance-covariance matrix of $C_x \hat{\tau}$ is $C_x A^- C_x' \sigma^2$. The structure of A^- defined in (3.16) motivates the structure of the set of contrast matrices, C_x, that is derived in the following section.

4.3 Contrast matrices for α_n-designs

By convention the vector of treatment parameters, τ, is constructed so that the parameters are subscripted lexicographically. However, an alternative ordering that exploits the cyclic nature of α_n-designs is now considered. A consequence of this ordering is that a set of contrast matrices, similar in structure to the generalized inverse matrix in (3.16), can be defined.

4.3.1 Contrast matrices for α_2-designs

Consider an experiment with two factors F_1 and F_2 at v_1 and v_2 levels respectively. Let the $v = v_1 v_2$ treatment combinations be set out in an α_2-design with r replicates and k plots per block.

The parameters $\tau_{i_1 i_2}$ in (4.1) can be written as

$$\tau_{i_1 i_2} = \overline{\tau} + (\overline{\tau}_{i_1 \cdot} - \overline{\tau}) + (\overline{\tau}_{\cdot i_2} - \overline{\tau}) + (\tau_{i_1 i_2} - \overline{\tau}_{i_1 \cdot} - \overline{\tau}_{\cdot i_2} + \overline{\tau}) \tag{4.2}$$

where $\overline{\tau}_{i_1 \cdot}$ and $\overline{\tau}_{\cdot i_2}$ are means averaged over the levels of factors F_2 and F_1 respectively, and $\overline{\tau}$ is the overall mean of the $\tau_{i_1 i_2}$ ($i_m = 0, 1, \ldots, v_m - 1;\ m = 1, 2$). This identity shows how each factorial treatment effect can be partitioned into orthogonal components corresponding to the main effects and interaction. Thus, the second and third terms on the right hand side of equation (4.2) represent the main effects of factors F_1 and F_2 respectively, and the final term the $F_1 F_2$ interaction.

In matrix notation, the set of equations given by (4.2) can be written as

$$\tau = C_{00}\tau + C_{10}\tau + C_{01}\tau + C_{11}\tau \tag{4.3}$$

where τ is a $v \times 1$ vector of the treatment parameters $\tau_{i_1 i_2}$, and where $C_{00} = K_v$ and C_{10}, C_{01} and C_{11} are $v \times v$ contrast matrices.

The objective now is to define a set of contrast matrices C_x ($x = x_1 x_2$), where $x_i = 1$ if factor F_i is present in the interaction and zero otherwise ($i = 1, 2$).

To simplify the discussion let the two factors F_1 and F_2 be at $v_1 = 6$ and $v_2 = 4$ levels respectively, and let the $v = 24$ treatment combinations be set out in an α_2-design with $r = 3$ replicates and $k = 4$ plots per block. Such a design is given in Table 3.3, where the $v = 24$ treatment combinations were divided into $k = 4$ mutually disjoint subsets of order $s = 6$, i.e.

$$\begin{aligned} S_1 &= \{00, 01, 10, 11, 20, 21\} \\ S_2 &= \{02, 03, 12, 13, 22, 23\} \\ S_3 &= \{30, 31, 40, 41, 50, 51\} \\ S_4 &= \{32, 33, 42, 43, 52, 53\} \end{aligned}$$

This partitioning is dependent on the factorisation of s, i.e. $s = s_1 s_2$ where $s_1 = 3$ and $s_2 = 2$.

Now, let τ be defined as in (4.3). Furthermore, let the treatment parameters in τ be subscripted by the ordered 2-tuples in S_1, followed by those in S_2, then S_3, and finally S_4, i.e.

$$\tau = (\tau_{00}\ \tau_{01}\ \tau_{10}\ \tau_{11}\ \tau_{20}\ \tau_{21} \mid \tau_{02}\ \tau_{03}\ \tau_{12}\ \tau_{13}\ \tau_{22}\ \tau_{23} \mid \tau_{30}\ \tau_{31}\ \tau_{40}\ \tau_{41}\ \tau_{50}\ \tau_{51} \mid \tau_{32}\ \tau_{33}\ \tau_{42}\ \tau_{43}\ \tau_{52}\ \tau_{53})' \tag{4.4}$$

The contrast matrices C_{10}, C_{01} and C_{11} must be defined with respect to the structure of τ in (4.4) so that they yield the main effects and interaction respectively. The block parameters, s_1 and s_2, play an integral role in their construction.

The ith row of $C_{10}\tau$ is $\overline{\tau}_{i_1\cdot} - \overline{\tau}$ so that C_{10} can be written as $C_{10} = M_{10} - C_{00}$, where $4M_{10}$ is given by the matrix in Figure 4.1 ($i_1 = 0, 1, \ldots, v_1 - 1$). Note, for example, that the first row of $4M_{10}$ has a 1 in each column corresponding to a treatment parameter in τ that has factor F_1 at level 0. Thus, pre-multiplying τ by M_{10} has the effect of averaging over the levels of factor F_2 when factor F_1 is at level i_1 ($i_1 = 0, 1, \ldots, 5$). It can be seen that

$$M_{10} = \frac{1}{4}\begin{pmatrix} R_1 & 0 \\ 0 & R_1 \end{pmatrix} = (1/4)I_2 \otimes R_1$$

where the 12×12 matrix R_1 can be written as

$$R_1 = \begin{pmatrix} R_2 & R_2 \\ R_2 & R_2 \end{pmatrix} = J_2 \otimes R_2$$

and where

$$R_2 = \begin{pmatrix} J_2 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_2 \end{pmatrix} = I_3 \otimes J_2$$

It follows that

$$M_{10} = (I_2 \otimes K_2) \otimes (I_3 \otimes K_2) \tag{4.5}$$

since $K_2 = (1/2)J_2$. The role of the identity matrices in (4.5) can be seen to have the effect of being an indicator of those parameters in τ which have factor F_1 at the i_1th level while the K_2 matrices average over the levels of factor F_2.

	τ_{00}	τ_{01}	τ_{10}	τ_{11}	τ_{20}	τ_{21}	τ_{02}	τ_{03}	τ_{12}	τ_{13}	τ_{22}	τ_{23}	τ_{30}	τ_{31}	τ_{40}	τ_{41}	τ_{50}	τ_{51}	τ_{32}	τ_{33}	τ_{42}	τ_{43}	τ_{52}	τ_{53}
τ_{00}	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{01}	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{10}	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{11}	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{20}	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
τ_{21}	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
τ_{02}	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{03}	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{12}	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{13}	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
τ_{22}	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
τ_{23}	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
τ_{30}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
τ_{31}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
τ_{40}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0
τ_{41}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0
τ_{50}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1
τ_{51}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1
τ_{32}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
τ_{33}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
τ_{42}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0
τ_{43}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0
τ_{52}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1
τ_{53}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1

Figure 4.1: The matrix $4M_{10}$

Similarly, the ith row of $C_{01}\tau$ is given by $\overline{\tau}_{\cdot i_2} - \overline{\tau}$ $(i_2 = 0, 1, \ldots, v_2 - 1)$. Thus, a contrast matrix for the main effect of factor F_2 at $v_2 = 4$ levels is given by $C_{01} = M_{01} - C_{00}$, where $6M_{01}$ is given by the matrix in Figure 4.2. The first row of the matrix $6M_{01}$ has a 1 in those columns that correspond with treatment parameters that have factor F_2 at level 0. Hence, pre-multiplying τ by M_{01} has the effect of averaging over the levels of factor F_1 when factor F_2 is at level i_2. It is now seen that

$$M_{01} = \frac{1}{6}\begin{pmatrix} R_1 & R_1 \\ R_1 & R_1 \end{pmatrix} = (1/6)J_2 \otimes R_1$$

where R_1 is now the 12×12 matrix given by

$$R_1 = \begin{pmatrix} R_2 & 0 \\ 0 & R_2 \end{pmatrix} = I_2 \otimes R_2$$

and

$$R_2 = \begin{pmatrix} I_2 & I_2 & I_2 \\ I_2 & I_2 & I_2 \\ I_2 & I_2 & I_2 \end{pmatrix} = J_3 \otimes I_2$$

It follows that

$$M_{01} = (K_2 \otimes I_2) \otimes (K_3 \otimes I_2) \tag{4.6}$$

since $K_{s_i} = (1/s_i)J_{s_i}$ for $i = 1, 2$.

	τ_{00}	τ_{01}	τ_{10}	τ_{11}	τ_{20}	τ_{21}	τ_{02}	τ_{03}	τ_{12}	τ_{13}	τ_{22}	τ_{23}	τ_{30}	τ_{31}	τ_{40}	τ_{41}	τ_{50}	τ_{51}	τ_{32}	τ_{33}	τ_{42}	τ_{43}	τ_{52}	τ_{53}
τ_{00}	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
τ_{01}	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
τ_{10}	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
τ_{11}	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
τ_{20}	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
τ_{21}	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
τ_{02}	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
τ_{03}	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
τ_{12}	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
τ_{13}	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
τ_{22}	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
τ_{23}	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
τ_{30}	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
τ_{31}	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
τ_{40}	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
τ_{41}	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
τ_{50}	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
τ_{51}	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
τ_{32}	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
τ_{33}	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
τ_{42}	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
τ_{43}	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1
τ_{52}	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
τ_{53}	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1

Figure 4.2: The matrix $6M_{01}$

From (4.2) the interaction component $\tau_{i_1 i_2} - \bar{\tau}_{i_1 .} - \bar{\tau}_{. i_2} + \bar{\tau}$ can be re-expressed to include the main effect of each factor, i.e.

$$\tau_{i_1 i_2} - (\bar{\tau}_{i_1 .} - \bar{\tau}) - (\bar{\tau}_{. i_2} - \bar{\tau}) - \bar{\tau} \tag{4.7}$$

In matrix notation, the set of equations given by (4.7) can be written as

$$C_{11}\tau = I_{24}\tau - C_{10}\tau - C_{01}\tau - C_{00}\tau \tag{4.8}$$

where C_{10} and C_{01} are obtained from (4.5) and (4.6) respectively, and C_{00} is the averaging matrix K_{24}. The identity matrix, I_{24}, can be expressed in terms of four matrix components, i.e.

$$M_{11} = (I_2 \otimes I_2) \otimes (I_3 \otimes I_2)$$

More generally, given the method of dividing the $v = v_1 v_2$ treatments into $k = k_1 k_2$ mutually disjoint subsets of order $s = s_1 s_2$, a contrast matrix for the main effect of factor F_1 is given by $C_{10} = M_{10} - C_{00}$, where

$$M_{10} = (I_{k_1} \otimes K_{k_2}) \otimes (I_{s_1} \otimes K_{s_2}) \tag{4.9}$$

and $C_{00} = (K_{k_1} \otimes K_{k_2}) \otimes (K_{s_1} \otimes K_{s_2})$. Hence, the matrix M_{10} averages over the levels of the second factor for each level of the first. A contrast matrix for the main effect of factor F_2 is given by $C_{01} = M_{01} - C_{00}$, where

$$M_{01} = (K_{k_1} \otimes I_{k_2}) \otimes (K_{s_1} \otimes I_{s_2}) \tag{4.10}$$

It averages over the levels of the first factor for each level of the second. The forms of M_{10} and M_{01} follow from the method in which the vector of treatment parameters is generated. Finally, from (4.8) it follows that a contrast matrix for the $F_1 F_2$ interaction is given by

$$C_{11} = I_{24} - C_{10} - C_{01} - C_{00} \tag{4.11}$$

with I_{24} replaced by the matrix $M_{11} = (I_{k_1} \otimes I_{k_2}) \otimes (I_{s_1} \otimes I_{s_2})$.

4.3.2 Contrast matrices for a generalized interaction

Suppose now that there is a third factor in the experiment so that the ith factor, F_i, is at v_i levels and $v_i = k_i s_i$ $(i = 1, 2, 3)$. Also, suppose that the experiment

is set out in an α_3-design. Let the treatment parameters $\tau_{j_1 j_2 j_3}$ in τ be ordered with respect to the sets $S_1, S_2, \ldots, S_k$, with the 3-tuples lexicographically ordered within subsets ($j_i = 0, 1, \ldots, v_i - 1$). Then, contrast matrices for the main effects of factors F_1 and F_2 and F_1F_2 interaction can be constructed from C_{10}, C_{01} and C_{11}, respectively, by including the averaging matrices K_{k_3} and K_{s_3} corresponding to the levels of factor F_3. Hence, contrast matrices for the main effects of factors F_1 and F_2 are respectively

$$C_{100} = (I_{k_1} \otimes K_{k_2} \otimes K_{k_3}) \otimes (I_{s_1} \otimes K_{s_2} \otimes K_{s_3}) - C_{000}$$

and

$$C_{010} = (K_{k_1} \otimes I_{k_2} \otimes K_{k_3}) \otimes (K_{s_1} \otimes I_{s_2} \otimes K_{s_3}) - C_{000}$$

where

$$C_{000} = (K_{k_1} \otimes K_{k_2} \otimes K_{k_3}) \otimes (K_{s_1} \otimes K_{s_2} \otimes K_{s_3})$$

is the overall averaging matrix. The first terms in C_{100} and C_{010} are the averaging matrices for the effects of interest and will be denoted M_{100} and M_{010} respectively. Notice that the identity matrices correspond to those factors that are present in the effect of interest. A contrast matrix for the F_1F_2 interaction is constructed in the same way, i.e.

$$C_{110} = (I_{k_1} \otimes I_{k_2} \otimes K_{k_3}) \otimes (I_{s_1} \otimes I_{s_2} \otimes K_{s_3}) - C_{100} - C_{010} - C_{000}$$

The identity matrices now correspond to the factors F_1 and F_2 which are present in the interaction. Hence, the matrix M_{110} averages over the levels of factor F_3 since it is absent from the interaction, while the remaining matrices effectively adjust for the main effects of factors F_1 and F_2 and the overall average effect. From the definitions of C_{100}, C_{010} and C_{000} given above, the contrast matrix C_{110} can be expressed entirely in terms of averaging matrices, $M_{x_1x_2x_3}$, and the overall averaging matrix, C_{000}, where $x_i = 1$ if factor F_i is present in the interaction and zero otherwise ($i = 1, 2, 3$), i.e.

$$C_{110} = M_{110} - M_{100} - M_{010} + C_{000}$$

Following in this light, the remaining contrast matrices for those effects in which factor F_3 *is* present are given by

$$\begin{aligned}
C_{001} &= M_{001} - C_{000} \\
C_{101} &= M_{101} - M_{100} - M_{001} + C_{000} \\
C_{011} &= M_{011} - M_{010} - M_{001} + C_{000} \\
C_{111} &= M_{111} - M_{110} - M_{101} - M_{011} + M_{100} + M_{010} + M_{001} - C_{000}
\end{aligned}$$

where the ith subscript of $C_{x_1x_2x_3}$ is 1 if factor F_i is present in the effect and 0 otherwise.

More generally, consider an α_n-design with $v = v_1v_2 \ldots v_n$ factorial treatment combinations arranged in $s = s_1s_2 \ldots s_n$ blocks of size $k = k_1k_2 \ldots k_n$. The vector of treatment parameters, τ, is constructed by subscripting the parameters with the ordered n-tuples in the mutually disjoint subsets $S_1, S_2, \ldots, S_k$ of the v factorial treatments. The subsets are generated by adding each n-tuple in $< s_1, s_2, \ldots, s_n >= \{b_1b_2 \ldots b_n : b_i \in < s_i >\}$ to every element of $Z_{s_1,s_2,\ldots,s_n} = \{a_1a_2 \ldots a_n : a_i \in Z_{s_i}\}$, where $< s_i >= \{0, s_i, \ldots, (k_i - 1)s_i\}$ and $Z_{s_i} = \{0, 1, \ldots, s_i - 1\}$ for $i = 1, 2, \ldots, n$. The averaging matrix for a generalized interaction effect, $F^x = F_1^{x_1}F_2^{x_2} \ldots F_n^{x_n}$, where $x_i = 1$ if the ith factor is present in the interaction and zero otherwise, is given by

$$M_x = Q_{xk} \otimes Q_{xs} \tag{4.12}$$

where

$$Q_{xp} = Q_{x_1p_1} \otimes Q_{x_2p_2} \otimes \cdots \otimes Q_{x_np_n} \tag{4.13}$$

and

$$Q_{x_ip_i} = \begin{cases} I_{p_i} & \text{if } x_i = 1 \\ K_{p_i} & \text{if } x_i = 0 \end{cases} \tag{4.14}$$

where $p_i = k_i$ or s_i $(i = 1, 2, \ldots n)$. A set of contrast matrices for any generalized interaction, F^x, is obtained by adjusting (4.12) for all lower order interactions containing the factors present in the effect of interest *and* the overall average effect. In general, therefore, a set of contrast matrices for any generalized interaction of an

α_n-design is given by

$$C_x = M_x - \sum_y C_y \tag{4.15}$$

where $\sum_y$ is the summation over all combinations of $y = y_1 y_2 \dots y_n$ such that $y_i = 0$ or x_i, except the combination $y = x$.

The relation for the C_x matrices defined by (4.15) is recursive and can be expressed as a linear combination of averaging matrices, i.e.

$$C_x = \sum_{\forall y} a_y M_y \tag{4.16}$$

where the summation $\sum_{\forall y}$ is over all combinations of y, *including* $y = x$. It can be shown by induction that the coefficients

$$a_y = (-1)^{\delta_x - \delta_y} \tag{4.17}$$

where δ_x is the number of factors present in the interaction and $\delta_y = \sum_{i=1}^{n} y_i$. Since $(-1)^{-1} = -1$ it follows that $(-1)^{-\delta_y} = (-1)^{\delta_y}$. Hence, from (4.16)

$$C_x = (-1)^{\delta_x} \sum_{\forall y} (-1)^{\delta_y} M_y \tag{4.18}$$

The C_x matrices form a complete binary set since they are symmetric ($C_x' = C_x$), mutually orthogonal ($C_x C_y = 0, x \neq y$), idempotent ($C_x^2 = C_x$) and satisfy $\sum_{\forall x} C_x = I$. Further, since C_x is idempotent then

$$\text{rank}(C_x) = \text{trace}(C_x) = \prod_{i=1}^{n} (v_i - 1)^{x_i} \tag{4.19}$$

In the following section, the set of contrast matrices given by (4.18) will be used to derive a general expression for the average efficiency factor of a generalized interaction effect for an α_n-design.

4.4 Average efficiency factor for a generalized interaction, E_x

Let C_x be a symmetric and idempotent contrast matrix for a generalized interaction $F^x = F_1^{x_1} F_2^{x_2} \dots F_n^{x_n}$, where $x = x_1 x_2 \dots x_n$ and $x_i = 1$ if factor F_i is present in the

interaction and $x_i = 0$ otherwise ($i = 1, 2, \ldots, n$). From (1.6) it follows that the average efficiency factor of any generalized interaction, F^x, can be written as

$$E_x = \frac{\text{trace}(C_x)}{r\text{trace}(C_x A^-)} \tag{4.20}$$

Now, let C_x be defined as in (4.18). Then, from (4.19) it follows that

$$E_x = \frac{\nu_x}{r\text{trace}(C_x A^-)} \tag{4.21}$$

where

$$\nu_x = \prod_{i=1}^{n} (v_i - 1)^{x_i} \tag{4.22}$$

is the number of degrees of freedom associated with the generalized interaction F^x.

A closed form mathematical expression for the average efficiency factor, E_x, of any generalized interaction F^x for an α_n-design will now be derived. First, the recursive relation for C_x given in (4.15) and the generalized inverse matrix A^- defined in (3.16) are substituted into trace($C_x A^-$). This yields an expression which is a summation over the trace of a Kronecker product of matrices of the form

$$Q_{uk} A^{*-}_{u_1 u_2 \ldots u_n} \otimes \left(\bigotimes_{i=1}^{n} Q_{us} \Gamma^*_{u_i} \right)$$

where $A^{*-}_{u_1 u_2 \ldots u_n}$ and $\Gamma^*_{u_i}$ are defined in (3.17) and (2.9) respectively, and where $u = u_1 u_2 \ldots u_n$. This trace is simplified by considering the values that trace($Q_{us} \Gamma^*_{u_i}$) can take for various values of u. Further simplification is achieved by separately considering the matrices $A^{*-}_{00\ldots0}$ and $A^{-1}_{u_1 u_2 \ldots u_n}$ for $u_1 + u_2 + \cdots + u_n > 0$. It is then shown that the contribution from any $A^{*-}_{u_1 u_2 \ldots u_n}$ matrix to the average efficiency factors is dependent on both the factors present in the effect of interest *and* the value of u. This then leads to the general expression for E_x given at the end of this section by equation (4.31).

The trace of $C_x A^-$ in (4.21) effectively sums over appropriate contrasts of the elements of each column of A^-. It will be shown that the effort required to compute this quantity can be reduced by averaging over appropriate elements of the $k \times k$ matrices $A^{*-}_{u_1 u_2 \ldots u_n}$ defined in (3.17). This result is a consequence of the conforming structures of the C_x matrices defined in the previous section and A^-.

From the definitions of A^- in (3.16) and the C_x matrices in (4.16) it follows that

$$C_x A^- = \sum_{\forall u} \sum_{\forall y} a_y \left(Q_{yk} A^{*-}_{u_1 u_2 \ldots u_n} \otimes Q_{ys} \Gamma^*_{u_i} \right)$$

where $\sum_{\forall_u}$ is the summation over all combinations of $u_1 u_2 \ldots u_n$ and $\sum_{\forall_y}$ is the summation over all combinations of $y_1 y_2 \ldots y_n$, and where a_y, Q_{yk} and Q_{ys} are defined in (4.17) and (4.13) respectively. Hence,

$$\text{trace}(C_x A^-) = \sum_{\forall_u} \sum_{\forall_y} a_y \text{trace} \left(Q_{yk} A^{*-}_{u_1 u_2 \ldots u_n} \right) \prod_{i=1}^{n} \text{trace} \left(Q_{ys} \Gamma^*_{u_i} \right)$$

From Lemma 2.1 it follows that

$$Q_{y_i s_i} \Gamma^*_{u_i} = \begin{cases} K_{s_i} & \text{if } u_i = 0 \\ \Gamma^*_{u_i} & \text{if } y_i = 1 \text{ and } u_i > 0 \\ 0 & \text{if } y_i = 0 \text{ and } u_i > 0 \end{cases}$$

Since $\text{trace}(\Gamma^*_{u_i}) = 1$ and $\Gamma^*_0 = K_{s_i}$, it follows from (4.14) that

$$\text{trace} \left(Q_{y_i s_i} \Gamma^*_{u_i} \right) = \begin{cases} 1 & \text{if } y_i = 1 \text{ or } u_i = 0 \\ 0 & \text{if } y_i = 0 \text{ and } u_i > 0 \end{cases} \tag{4.23}$$

It can be verified that $\text{trace}(Q_{y_i s_i} \Gamma^*_{u_i}) = y_i^{u_i}$ satisfies (4.23). Hence,

$$\text{trace}(C_x A^-) = \sum_{\forall_u} f(u, x) \tag{4.24}$$

where

$$f(u, x) = \sum_{\forall_y} a_y \text{trace} \left(Q_{yk} A^{*-}_{u_1 u_2 \ldots u_n} \right) \prod_{i=1}^{n} y_i^{u_i} \tag{4.25}$$

Note that the coefficient matrix, Q_{yk}, of $A^{*-}_{u_1 u_2 \ldots u_n}$ is that component of the averaging matrix M_x, defined by (4.12), dependent on only the parameters k_i ($i = 1, 2, \ldots, n$). If the ith factor is absent from the generalized interaction effect (i.e. $x_i = 0$), then those $A^{*-1}_{u_1 u_2 \ldots u_n}$ matrices with $u_i > 0$ do not contribute to $\text{trace}(C_x A^-)$ and hence, do not contribute to the corresponding average efficiency factor. This result is due to the product $\prod_{i=1}^{n} y_i^{u_i}$ taking only the values 0 or 1.

A further simplification of $\text{trace}(C_x A^-)$ is now possible. For instance, since $A^{*-}_{00\ldots0} = I_k / r$ its contribution to $\text{trace}(C_x A^-)$ is known and, again by induction, it can be shown to be

$$\frac{1}{r} \sum_{\forall_y} a_y \prod_{i=1}^{n} \text{trace} \left(Q_{y_i k_i} \right) = \frac{1}{r} \kappa_x \tag{4.26}$$

where

$$\kappa_x = \prod_{i=1}^{n}(k_i - 1)^{x_i} \tag{4.27}$$

It now follows that

$$\text{trace}(C_x A^-) = \frac{1}{r}\kappa_x + \sum_u f(u, x) \tag{4.28}$$

where

$$f(u,x) = (-1)^{\delta_x} \sum_{\forall_y} (-1)^{\delta_y} \text{trace}\left(Q_{yk} A^{*-1}_{u_1 u_2 \ldots u_n}\right) \prod_{i=1}^{n} {y_i}^{u_i} \tag{4.29}$$

and where $\sum_u$ is the summation over all combinations of $u_1 u_2 \ldots u_n$, except $00\ldots0$.

First, consider an α_2-design with treatment factors F_1 and F_2 at v_1 and v_2 levels, respectively, consisting of r replicates arranged in $s = s_1 s_2$ blocks of size $k = k_1 k_2$. From (4.21), (4.28) and (4.29) it follows that the average efficiency factor $E_{x_1 x_2}$ for an α_2-design is given by

$$E_{x_1 x_2} = \frac{(v_1 - 1)^{x_1}(v_2 - 1)^{x_2}}{r\,\text{trace}(C_{x_1 x_2} A^-)}$$

where

$$\text{trace}(C_{x_1 x_2} A^-) = \frac{1}{r}(k_1 - 1)^{x_1}(k_2 - 1)^{x_2} + \sum_{\substack{u_1=0 \\ (u_1+u_2 \neq 0)}}^{s_1-1} \sum_{u_2=0}^{s_2-1} f(u_1 u_2, x_1 x_2)$$

and

$$f(u_1 u_2, x_1 x_2) = (-1)^{x_1+x_2} \sum_{y_1=0}^{x_1} \sum_{y_2=0}^{x_2} (-1)^{y_1+y_2} \text{trace}\Big[\left(Q_{y_1 k_1} \otimes Q_{y_2 k_2}\right) A^{*-1}_{u_1 u_2}\Big] {y_1}^{u_1} {y_2}^{u_2}$$

The average efficiency factor for the factor F_1 main effect is obtained when $x_1 = 1$ and $x_2 = 0$, i.e.

$$E_{10} = \frac{(v_1 - 1)}{r\,\text{trace}(C_{10} A^-)}$$

In this case

$$\text{trace}(C_{10} A^-) = \frac{1}{r}(k_1 - 1) + \sum_{u_1=1}^{s_1-1} \text{trace}\Big[(Q_{1k_1} \otimes Q_{0k_2}) A^{*-1}_{u_1 0}\Big]$$

since $f(u_1 u_2, 10)$ contributes to $\text{trace}(C_{10} A^-)$ only when $u_2 = 0$. From (4.14) it follows that the average efficiency factor for the F_1 main effect is given by

$$E_{10} = \frac{(v_1 - 1)}{(k_1 - 1) + r \sum_{u_1=1}^{s_1-1} \text{trace}\Big[(I_{k_1} \otimes K_{k_2}) A^{*-1}_{u_1 0}\Big]}$$

The average efficiency factor for the F_2 main effect is obtained when $x_1 = 0$ and $x_2 = 1$, i.e.

$$E_{01} = \frac{(v_2 - 1)}{r\text{trace}(C_{01}A^-)}$$

Now,

$$\text{trace}(C_{01}A^-) = \frac{1}{r}(k_2 - 1) + \sum_{u_2=1}^{s_2-1} \text{trace}\left[(Q_{0k_1} \otimes Q_{1k_2})A_{0u_2}^{*-1}\right]$$

since $f(u_1u_2, 01)$ contributes to $\text{trace}(C_{01}A^-)$ only when $u_1 = 0$. Hence,

$$E_{01} = \frac{(v_2 - 1)}{(k_2 - 1) + r\sum_{u_2=1}^{s_2-1} \text{trace}\left[(K_{k_1} \otimes I_{k_2})A_{0u_2}^{*-1}\right]}$$

Finally, the average efficiency factor for the F_1F_2 interaction effect is obtained when $x_1 = x_2 = 1$, i.e.

$$E_{11} = \frac{(v_1 - 1)(v_2 - 1)}{r\text{trace}(C_{11}A^-)}$$

The contribution to $\text{trace}(C_{11}A^-)$ from (4.29) is now

$$\begin{aligned} f(u_1u_2, 11) &= \text{trace}\left[(K_{k_1} \otimes K_{k_2})\, A_{u_1u_2}^{*-1}\right] 0^{u_1+u_2} - \text{trace}\left[(K_{k_1} \otimes I_{k_2})\, A_{u_1u_2}^{*-1}\right] 0^{u_1} 1^{u_2} \\ &\quad -\text{trace}\left[(I_{k_1} \otimes K_{k_2})\, A_{u_1u_2}^{*-1}\right] 1^{u_1} 0^{u_2} + \text{trace}\left[(I_{k_1} \otimes I_{k_2})\, A_{u_1u_2}^{*-1}\right] 1^{u_1+u_2} \end{aligned}$$

Since the summation in $\text{trace}(C_{11}A^-)$ is over all combinations of u_1 and u_2, except $u_1 = u_2 = 0$, the first term in $f(u_1u_2, 11)$ is zero. The second term makes a contribution to $\text{trace}(C_{11}A^-)$ only when $u_2 = 0$; similarly, the third term contributes when $u_1 = 0$. It follows that

$$\begin{aligned} \text{trace}(C_{11}A^-) &= \frac{1}{r}(k_1 - 1)(k_2 - 1) - \sum_{u_1=1}^{s_1-1} \text{trace}\left[(I_{k_1} \otimes K_{k_2})\, A_{u_10}^{*-1}\right] \\ &\quad - \sum_{u_2=1}^{s_2-1} \text{trace}\left[(K_{k_1} \otimes I_{k_2})\, A_{0u_2}^{*-1}\right] \\ &\quad + \sum_{\substack{u_1=0 \\ (u_1+u_2\neq 0)}}^{s_1-1} \sum_{u_2=0}^{s_2-1} \text{trace}\left[(I_{k_1} \otimes I_{k_2})\, A_{u_1u_2}^{*-1}\right] \end{aligned}$$

Expanding the last term in the above expression gives

$$\begin{aligned} \text{trace}(C_{11}A^-) &= \frac{1}{r}(k_1 - 1)(k_2 - 1) - \sum_{u_1=1}^{s_1-1} \text{trace}\left[(I_{k_1} \otimes K_{k_2})\, A_{u_10}^{*-1}\right] \\ &\quad - \sum_{u_2=1}^{s_2-1} \text{trace}\left[(K_{k_1} \otimes I_{k_2})\, A_{0u_2}^{*-1}\right] + \sum_{u_1=1}^{s_1-1} \text{trace}\left[(I_{k_1} \otimes I_{k_2})\, A_{u_10}^{*-1}\right] \\ &\quad + \sum_{u_2=1}^{s_2-1} \text{trace}\left[(I_{k_1} \otimes I_{k_2})\, A_{0u_2}^{*-1}\right] + \sum_{u_1=1}^{s_1-1} \sum_{u_2=1}^{s_2-1} \text{trace}\left[(I_{k_1} \otimes I_{k_2})\, A_{u_1u_2}^{*-1}\right] \end{aligned}$$

Those terms dependent on only u_1 and, separately, on only u_2 can now be factorised so that the average efficiency factor for the F_1F_2 interaction can be written as

$$E_{11} = \frac{(v_1 - 1)(v_2 - 1)}{(k_1 - 1)(k_2 - 1) + r\Delta}$$

where

$$\begin{aligned}\Delta &= \sum_{u_1=1}^{s_1-1}\sum_{u_2=1}^{s_2-1} \text{trace}\left[(I_{k_1} \otimes I_{k_2})A^{*-1}_{u_1u_2}\right] + \sum_{u_1=1}^{s_1-1} \text{trace}\left\{[I_{k_1} \otimes (I_{k_2} - K_{k_2})]A^{*-1}_{u_10}\right\} \\ &\quad + \sum_{u_2=1}^{s_2-1} \text{trace}\left\{[(I_{k_1} - K_{k_1}) \otimes I_{k_2}]A^{*-1}_{0u_2}\right\}\end{aligned}$$

Unlike the average efficiency factor for the F_1 main effect, E_{10}, and F_2 main effect, E_{01}, the average efficiency factor for the F_1F_2 interaction, E_{11}, uses information from *all* the $A^{*-}_{u_1u_2}$ matrices. In particular, notice that Δ has been partitioned into the contributions from $A^{*-1}_{u_1u_2}$ when $u_1 > 0$ and $u_2 > 0$, $u_1 > 0$ and $u_2 = 0$, and $u_1 = 0$ and $u_2 > 0$. Also, the coefficient matrices $I_{k_1} \otimes (I_{k_2} - K_{k_2})$, when $u_2 = 0$, and $(I_{k_1} - K_{k_1}) \otimes I_{k_2}$, when $u_1 = 0$, are necessarily different to those in E_{10} and E_{01}, respectively, since now both factors are present in the interaction.

Now, let

$$L_{u_ix_i} = \begin{cases} I_{k_i} - K_{k_i} & \text{if } x_i = 1,\ u_i = 0 \\ I_{k_i} & \text{if } x_i = 1,\ u_i > 0 \\ K_{k_i} & \text{if } x_i = 0,\ u_i = 0 \\ 0 & \text{if } x_i = 0,\ u_i > 0 \end{cases} \tag{4.30}$$

for $i = 1, 2$. Then, the average efficiency factor of a generalized interaction, F^x, for an α_2-design can be written as

$$E_{x_1x_2} = \frac{(v_1 - 1)^{x_1}(v_2 - 1)^{x_2}}{(k_1 - 1)^{x_1}(k_2 - 1)^{x_2} + r\sum\limits_{\substack{u_1=0 \\ (u_1+u_2\neq 0)}}^{s_1-1}\sum\limits_{u_2=0}^{s_2-1} \text{trace}\left[\left(L_{u_1x_1} \otimes L_{u_2x_2}\right) A^{*-1}_{u_1u_2}\right]}$$

From (4.30), the coefficient matrix $L_{x_1u_1} \otimes L_{x_2u_2}$ in $E_{x_1x_2}$ shows that the contributions from A^- to the average within block contrast variance for a given main effect or interaction is dependent on both the value of u_i and the factors present in the effect. The subscripts u_1 and u_2 are used to select the contributing $A^{*-1}_{u_1u_2}$ matrices. The coefficient matrices, $L_{u_1x_1} \otimes L_{u_2x_2}$, assume the role of averaging over appropriate elements of $A^{*-1}_{u_1u_2}$ for the given effect and are independent of s_i.

The contributions from A^- can be partitioned into those from A^{*-}_{00} and those from $A^{*-1}_{u_1u_2}$ for $u_1 + u_2 \neq 0$. When both u_1 and u_2 are zero, the contribution to the average contrast variance is given by the trace of the coefficient matrix, $L_{u_1x_1} \otimes L_{u_2x_2}$, divided by r, as was shown by (4.26) and (4.27). When u_1 and u_2 do not sum to zero, the contribution from A^- is dependent on those factors that are present in the effect. The presence or absence of a factor influences both those $A^{*-1}_{u_1u_2}$ matrices that contribute to the average within block contrast variance *and* the coefficient matrices that operate on them. For example, only the $A^{*-1}_{u_10}$ matrices contribute to the average efficiency factor for the F_1 main effect, while only the $A^{*-1}_{0u_2}$ matrices contribute to the average efficiency factor for the F_2 main effect.

In general, the average efficiency factor of a generalized interaction, F^x, for an α_n-design is given by

$$E_x = \frac{\nu_x}{\kappa_x + r \sum_{u_x} \text{trace}\left(L_{ux} A^{*-1}_{u_1u_2\ldots u_n}\right)} \tag{4.31}$$

where $\sum_u$ is the summation over all combinations of $u_1u_2\ldots u_n$, except the combination $00\ldots0$, and $L_{ux} = \bigotimes_{i=1}^{n} L_{u_ix_i}$, and where ν_x, κ_x and $L_{u_ix_i}$ are defined in (4.22), (4.27) and (4.30) respectively. This result can be verified by induction.

4.5 Calculating E_x

The average efficiency factor of a generalized interaction, F^x, for an α_n-design can be computed recursively. First, consider the recursive relation for the contrast matrix C_x defined in (4.15), namely

$$C_x = M_x - \sum_y C_y \tag{4.32}$$

It immediately follows that

$$\text{trace}(C_xA^-) = T_x/r - \textstyle\sum_{y_x} \text{trace}(C_yA^-) \tag{4.33}$$

where

$$T_x/r = \text{trace}(M_xA^-) - \text{trace}(C_{00\ldots0}A^-) \tag{4.34}$$

and where $\sum_{y_x}$ is the summation over all n-tuples $y_1y_2\ldots y_n$ such that $y_i = 0$ or x_i, excluding $y = x$ and $y = 00\ldots0$. Equation (4.33) shows that the trace of the matrix

product $C_x A^-$ can also be obtained recursively and consequently, substantially reduces the number of matrix operations required to compute E_x. Now, substituting (4.33) into (4.21) yields the following recursive relation for computing the average efficiency factor of a generalized interaction F^x, i.e.

$$\nu_x E_x^{-1} = T_x - \sum_{y_x} \nu_y E_y^{-1} \tag{4.35}$$

Now consider equation (4.34). From (4.12), (3.14) and (3.15) it follows that

$$\begin{aligned} \text{trace}(M_x A^-) &= \prod_{i=1}^{n} \text{trace}(Q_{x_i k_i}) \text{trace}(Q_{x_i s_i} \Gamma_0^*) \\ &\quad + \sum_{u} \text{trace}(Q_{xk} A^{*-1}_{u_1 u_2 \ldots u_n}) \prod_{i=1}^{n} \text{trace}(Q_{x_i s_i} \Gamma^*_{u_i}) \end{aligned}$$

where $\sum_u$ is the summation over all combinations of $u_1 u_2 \ldots u_n$ excluding $u_1 = u_2 = \cdots = u_n = 0$, and $Q_{xk} = \bigotimes_{i=1}^{n} Q_{x_i k_i}$ where $Q_{x_i k_i}$ and $Q_{x_i s_i}$ are defined in (4.14). From (4.14) and (4.23) this simplifies to

$$\text{trace}(M_x A^-) = \frac{1}{r} k_x + \sum_{u} \text{trace}(Q_{xk} A^{*-1}_{u_1 u_2 \ldots u_n}) \prod_{i=1}^{n} {x_i}^{u_i}$$

where $k_x = \prod_{i=1}^{n} k_i^{x_i}$. The product term $\prod_{i=1}^{n} {x_i}^{u_i}$ is non-zero when the ith factor is present in the interaction or $u_i = 0$. Finally, from (4.28) and (4.29) it can be shown that $\text{trace}(C_{00\ldots0} A^-) = 1/r$. Hence, T_x in (4.35) is given by

$$T_x = \frac{1}{r}(k_x - 1) + \sum_{u_x} \text{trace}(Q_{ux} A^{*-1}_{u_1 u_2 \ldots u_n}) \tag{4.36}$$

where $\sum_{u_x}$ is defined as in (4.31) with the added constraint that u_i is fixed at zero if the ith factor is absent from the interaction.

As an example, for $n = 3$ the average efficiency factor of the main effect F_1 is given by

$$E_{100} = \frac{v_1 - 1}{(k_1 - 1) + r \sum_{u_1=1}^{s_1 - 1} \text{trace}\left[(I_{k_1} \otimes K_{k_2} \otimes K_{k_3}) A^{*-1}_{u_1 00} \right]}$$

showing that the matrices $A^{*-1}_{u_1 00}$ are averaged over the levels of the second and third factors. Similarly, the average efficiency factor of the interaction of F_1 and F_3 is given by E_{101} where

$$(v_1 - 1)(v_3 - 1) E_{101}^{-1} = T_{101} - (v_1 - 1) E_{100}^{-1} - (v_3 - 1) E_{001}^{-1}$$

and where

$$T_{101} = \frac{1}{r}(k_1 k_3 - 1) + \sum_{\substack{u_1=0 \\ (u_1+u_3 \neq 0)}}^{s_1-1} \sum_{u_3=0}^{s_3-1} \text{trace}\left[(I_{k_1} \otimes K_{k_2} \otimes I_{k_3}) A^{*-1}_{u_1 0 u_3}\right]$$

This example shows that for α_n-designs the calculation of the average efficiency factors of any generalized interaction is straightforward given the inverses of the $k \times k$ Hermitian matrices $A^{*-1}_{u_1 u_2 \ldots u_n}$. In view of (4.35) the average efficiency factors for main effects should be calculated first, then those for two-factor interactions, then three-factor interactions, and so on.

Calculating E_x need not involve any operations on complex matrices. To illustrate, let

$$A^{*-1}_u = Y_{1u} + iY_{2u}$$

where Y_{1u} and Y_{2u} are $k \times k$ matrices with real entries and $i^2 = -1$, and where $u = u_1 u_2 \ldots u_n$. Then,

$$\text{trace}(L_{ux} A^{*-1}) = \text{trace}(L_{ux} Y_{1u}) + i\text{trace}(L_{ux} Y_{2u})$$

where L_{ux} is defined as in (4.31). It will now be shown that there is no contribution from the complex part of A^{*-1} to $\text{trace}(L_{ux} A^{*-1})$, i.e. $\text{trace}(L_{ux} Y_{2u}) = 0$.

Since L_{ux} is idempotent and the trace of matrix products is invariant under any cyclic permutation of the matrices, it follows that

$$\text{trace}(L_{ux} Y_{2u}) = \text{trace}(L'_{ux} Y_{2u} L_{ux})$$

Further, since interest lies in the trace of the matrix product $L'_{ux} Y_{2u} L_{ux}$, only its diagonal elements must be computed. Letting ℓ_{pq} represent the (p, q)th element of L_{ux}, the qth element along the leading diagonal of $L'_{ux} Y_{2u} L_{ux}$ is given by

$$(L'_{ux} Y_{2u} L_{ux})_q = \ell'_q Y_{2u} \ell_q$$

where $\ell_q = (\ell_{1q}, \ell_{2q}, \ldots, \ell_{kq})'$. Hence,

$$\text{trace}\,(L'_{ux} Y_{2u} L_{ux}) = \sum_{q=1}^{k} \ell'_q Y_{2u} \ell_q$$

Now, since Y_{2u} is skew-symmetric it follows that $\ell'_q Y_{2u} \ell_q = 0$ for all $k \times 1$ vectors ℓ_q that span the k-dimensional vector space $\mathbb{R}^k$; see Definition B.5 and Theorem B.7

in Appendix B. Hence, $\text{trace}(L'_{ux}Y_{2u}L_{ux}) = 0$, i.e. the complex part of $A^{*-1}_{u_1u_2\ldots u_n}$ makes no contribution to $\text{trace}(L'_{ux}Y_{2u}L_{ux})$. Thus, the $A^{*-1}_{u_1u_2\ldots u_n}$ matrices in E_x and T_x defined in (4.31) and (4.36), respectively, can be substituted with

$$Y_{1u} = (X_{1u} + X_{2u}X_{1u}^{-1}X_{2u})^{-1}$$

where the $k \times k$ matrices X_{1u} and X_{2u} are the symmetric and skew-symmetric parts, respectively, of $A^*_{u_1u_2\ldots u_n}$, i.e. $A^*_{u_1u_2\ldots u_n} = X_{1u} + iX_{2u}$.

4.6 Inter-effect orthogonality

In section 1.4 it was stated that an important objective in developing methodologies for the construction of factorial designs is that they have the property of orthogonal factorial structure (OFS). From Theorem 1.1, a sufficient condition for a connected block design to admit orthogonal factorial structure is that the information matrix, A, and contrast matrix, C_x, commute for all $x \in \{x_1x_2\ldots x_n : x_i \in \{0,1\} \text{ and } \sum_i x_i \neq 0 \text{ for } i = 1,2,\ldots,n\}$. The following counterexample will show that α_n-designs do not, in general, admit OFS.

Consider the α_2-design in Table 3.3 with $v_1 = 6$, $v_2 = 2$, $k_1 = k_2 = 2$, $s_1 = 3$, $s_2 = 2$ and $r = 3$. From (3.14), the information matrix for this design is given by

$$A = \sum_{u_1=0}^{2}\sum_{u_2=0}^{1} A^*_{u_1u_2} \otimes \Gamma^*_{u_1} \otimes \Gamma^*_{u_2}$$

and from (4.15) the contrast matrix for the factor F_1 main effect is given by

$$C_{10} = (I_2 \otimes K_2) \otimes (I_3 \otimes K_2) - (K_2 \otimes K_2) \otimes (K_3 \otimes K_2)$$

It will now be shown that the α_2-design in Table 3.3 does not have OFS by showing that $AC_{10} \neq C_{10}A$, i.e. A and C_{10} do not commute. Pre-multiplying A by C_{10}, after simplification, yields

$$C_{10}A = X - \sum_{u_1=1}^{2} P_{22}A^*_{u_10} \otimes \Gamma^*_{u_1} \otimes K_2 \tag{4.37}$$

while post-multiplying A by C_{10} yields

$$AC_{10} = X - \sum_{u_1=1}^{2} A^*_{u_10}P_{22} \otimes \Gamma^*_{u_1} \otimes K_2 \tag{4.38}$$

where

$$X = \frac{1}{3}(I_2 - K_2) \otimes K_2 \otimes K_3 \otimes K_2$$

and

$$P_{22} = I_2 \otimes K_2$$

Thus, from (4.37) and (4.38) it follows that $C_{10}A = AC_{10}$ if $P_{22}A^*_{u_10} = A^*_{u_10}P_{22}$.

From (3.15) it follows that

$$A^*_{u_10} = A_{00} + A_{00} + \exp(2\pi u_1/3)(A_{10} + A_{11}) + \exp(4\pi u_1/3)(A_{20} + A_{21})$$

so that from (3.12)

$$A^*_{10} = \frac{1}{8}(Y + i\sqrt{3}Z) \tag{4.39}$$

and

$$A^*_{20} = \frac{1}{8}(Y - i\sqrt{3}Z) \tag{4.40}$$

where

$$Y = 24I_4 - 2(B_{00} + B_{01}) - (B_{10} + B_{11} + B_{20} + B_{21})$$

and

$$Z = B_{10} + B_{11} - B_{20} - B_{21}$$

Substituting (4.39) and (4.40) into (4.37) gives

$$C_{10}A = X - \frac{1}{8}\left[P_{22}Y \otimes (\Gamma^*_1 + \Gamma^*_2) + i\sqrt{3}P_{22}Z \otimes (\Gamma^*_1 - \Gamma^*_2)\right] \otimes K_2$$

Similarly, substituting (4.39) and (4.40) into (4.38) gives

$$AC_{10} = X - \frac{1}{8}\left[YP_{22} \otimes (\Gamma^*_1 + \Gamma^*_2) + i\sqrt{3}ZP_{22} \otimes (\Gamma^*_1 - \Gamma^*_2)\right] \otimes K_2$$

Hence, $P_{22}A^*_{u_10} = A^*_{u_10}P_{22}$ if $P_{22}Y = YP_{22}$ *and* $P_{22}Z = ZP_{22}$. From the treatment concurrence matrix in Figure 3.1 it follows that

$$P_{22}Y = I_2 \otimes 21J_2 = YP_{22}$$

i.e. Y and P_{22} commute. However, P_{22} and Z do *not* commute since

$$P_{22}Z = \begin{pmatrix} 1 & -1 & -2 & 0 \\ 1 & -1 & -2 & 0 \\ 2 & 0 & -1 & 1 \\ 2 & 0 & -1 & 1 \end{pmatrix} \neq \begin{pmatrix} -1 & -1 & -2 & -2 \\ 1 & 1 & 0 & 0 \\ 2 & 2 & 1 & 1 \\ 0 & 0 & -1 & -1 \end{pmatrix} = ZP_{22}$$

It follows, therefore, that C_{10} and A do not commute, i.e. $C_{10}A \neq AC_{10}$. Thus, α_n-designs do not, in general, admit orthogonal factorial structure.

While the property of OFS is an attractive one it cannot always be achieved. In the next chapter it will be shown that there is a subset of α_n-designs which are known to admit OFS if r is a multiple of k.

Chapter 5

Resolvable n-cyclic designs

5.1 Introduction

The aim of this thesis has been to identify and define a family of resolvable incomplete block designs for factorial experiments. Early investigations included the class of n-dimensional cyclic designs, called n-cyclic designs, known to be suitable as non-resolvable designs for factorial experiments. These designs possess the important property of orthogonal factorial structure, and the precision with which main effects and interactions can be estimated is readily determined.

Using cyclic group theory, a sizeable subset of *resolvable* n-cyclic designs was identified. While efficient designs are contained in this subset, these require the number of replicates to equal the block size. However, the resources available for carrying out an experiment are often limited so that constraints are placed on the number of replicates. Consequently, reduced sets obtained by deleting one or more complete replicates from the full n-cyclic design were investigated.

The reduced n-cyclic sets did not preserve the orthogonal factorial structure of the full design. However, when the rows and columns of the treatment concurrence matrix were rearranged to correspond with the ordered treatments within the subgroup and cosets upon which their construction is based, the familiar structure of the α-designs of Patterson and Williams (1976) emerged. Hence, it was through investigating resolvable n-cyclic designs that the method for constructing α_n-designs was identified.

In this chapter, the family of n-cyclic designs will be examined. It will first be shown, in section 5.2, how the 1-dimensional cyclic designs can be set out in replicates. In section 5.3 it is shown how these results extend to n-cyclic designs. It is further shown that these resolvable n-cyclic designs are α_n-designs. Their properties are considered in the context of factorial experiments in section 5.4. Finally, in section 5.5, reducing the size of the design by deleting replicates from the full n-cyclic design is considered.

This work has been published in the *Journal of Statistical Planning and Inference*; see Appendix C.

5.2 Cyclic designs

Cyclic designs are incomplete block designs obtained by a cyclic development of one or more initial blocks whose elements are from the set Z_v. For example, a cyclic design for $v = 24$ treatments arranged in $b = 24$ blocks of size $k = 3$ generated from the initial block (0 1 5) is shown in Table 5.1, where columns represent blocks in the design. Each block in the design is obtained by adding 1 to each element in the previous block, reducing modulo 24 when necessary.

Table 5.1: Cyclic design for $v = 24$ treatments in blocks of size $k = 3$.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	0
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	0	1	2	3	4

David (1967) showed that the blocks of a cyclic design can be rearranged to yield a resolvable layout if every element in the set Z_k is represented when the treatments in the initial block are reduced modulo k. Since the initial block (0 1 5) of the cyclic design in Table 5.1 satisfies this requirement its blocks can be rearranged into $r = 3$ replicates, each consisting of eight blocks, as shown in Table 5.2. The jth replicate in the resolvable layout is formed by grouping together the blocks $b_j, b_{j+3}, \ldots, b_{j+(s-1)3}$, where b_ℓ is the ℓth block in the cyclic design ($j = 1, 2, 3; \ell = 1, 2, \ldots, 24$).

Each block of the design in Table 5.2 contains one element from each of

Table 5.2: Resolvable arrangement of cyclic design in Table 5.1

Replicate 1	Replicate 2	Replicate 3
0 3 6 9 12 15 18 21	1 4 7 10 13 16 19 22	2 5 8 11 14 17 20 23
1 4 7 10 13 16 19 22	2 5 8 11 14 17 20 23	3 6 9 12 15 18 21 0
5 8 11 14 17 20 23 2	6 9 12 15 18 21 0 3	7 10 13 16 19 22 1 4

the sets $S = \{0, 3, 6, 9, 12, 15, 18, 21\}$, $C_1 = \{1, 4, 7, 10, 13, 16, 19, 22\}$ and $C_2 = \{2, 5, 8, 11, 14, 17, 20, 23\}$, where S is the subgroup of Z_{24} generated by $k = 3$ and C_1 and C_2 are its cosets. Notice that the blocks in the first replicate are generated by adding each of the elements in S in turn to every element in the initial block, reducing modulo 24 when necessary. Similarly, the blocks in replicates 2 and 3 are generated by adding each of the elements in C_1 and C_2, respectively, to the elements in the initial block. Furthermore, note that any initial block that comprises one element from S and one from each of C_1 and C_2 will satisfy the criterion given by David for a resolvable cyclic design.

In general, suppose that the number of treatments v is divisible by the block size k. Then, there exists a subgroup $S = \{0, k, \ldots, (s-1)k\}$ of Z_v, closed under addition modulo v, whose $k-1$ cosets are $C_i = \{i, k+i, \ldots, (s-1)k+i\}$ for $i = 1, 2, \ldots, k-1$. A resolvable cyclic design with $r = k$ replicates can be generated from any initial block comprising one element from S and one from each of its cosets. The first replicate is generated by adding each element in S to every element in the initial block. The $(i+1)$th replicate is generated by adding each element in C_i in turn to every element in the initial block, reducing modulo v when necessary.

When v and k are *not* relatively prime partial resolvable cyclic designs with $r < k$ can be constructed. For example, consider an experiment with $v = 12$ treatments arranged in blocks of size $k = 4$. A resolvable cyclic design can be generated from the initial block (0 5 6 11) since it comprises one element from the subgroup $S = \{0, 4, 8\}$ and one from each of its cosets $C_1 = \{1, 5, 9\}$, $C_2 = \{2, 6, 10\}$ and $C_3 = \{3, 7, 11\}$. However, $d^* = 2$ divides both v and k so that another subgroup of Z_{12} is given by $S^* = \{0, 6\}$. Note that the initial block is made up of S^* and its coset C_5^*. Hence, adding each of the elements in S, C_1, C_2 and C_3 to those in the

initial block yields the design with only two distinct replicates shown in Table 5.3. The first replicate can be obtained either by adding each of the elements in S *or* each of the elements in C_2 to the initial block. This is because both S and C_2 are comprised of one element from each of S^*, C_2^* and C_4^*. Similarly, C_1 and C_3 are comprised of one element from each of C_1^*, C_3^* and C_5^* so that addition of each of the elements in either of these sets to those in the initial block will yield replicate 2.

Table 5.3: Resolvable cyclic design for $v = 12$, $k = 4$ and $r = 2$.

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

In general, suppose there exists an integer $d^* > 1$ such that $p = v/d^*$. Then there exists a subgroup $S^* = \{0, p, \ldots, (d^* - 1)p\}$ with cosets $C_m^* = \{m, p + m, \ldots, (d^* - 1)p + m\}$ for $m = 1, 2, \ldots, p - 1$. Now suppose that the initial block is formed from *all* the elements in S^* and $(k/d^*) - 1$ of its cosets. Provided this initial block comprises one element from the subgroup S and one element from each of its coset $C_1, C_2, \ldots, C_{k-1}$ then the partial cyclic set will be resolvable.

Partial resolvable cyclic designs cannot always be constructed. For instance, consider an experiment with $v = 24$ treatments in blocks of size $k = 4$. To generate a resolvable cyclic design with $r = 2$, then $d^* = 2$ and hence $S^* = \{0, 12\}$. However, since $S = \{0, 4, 8, 12, 16, 20\}$ any initial block comprising all the elements from S^* will contain two elements from S. Cyclic designs generated from any such initial block will *not* be resolvable.

5.3 n-Cyclic designs

Cyclic designs are obtained by a cyclic development of one or more initial blocks. Now suppose that the number of treatments in an experiment can be factorised as $v = v_1 v_2 \ldots v_n$ so that each treatment can be represented by an n-tuple in $Z_{v_1, v_2, \ldots, v_n}$.

If cyclic development is carried out on each digit in turn, the resulting design is called an *n-cyclic* design.

Resolvable n-cyclic designs can be constructed using a straightforward extension of the method used in the previous section. For example, consider the design of an experiment with $r = 3$ replicates of $v = 24$ treatments arranged in blocks of size $k = 3$. One possible factorisation of $v = 24$ is $v_1 = 6$ and $v_2 = 4$ so that the only possible factorisation of k satisfying $v_i/k_i \in \mathbb{Z}^+$ is $k_1 = 3$ and $k_2 = 1$. A subgroup of $Z_{6,4}$ is given by $S =< 3, 1 >= \{00, 01, 02, 03, 30, 31, 32, 33\}$. Its cosets are $C_1 = \{10, 11, 12, 13, 40, 41, 42, 43\}$ and $C_2 = \{20, 21, 22, 23, 50, 51, 52, 53\}$. They are obtained by adding the non-identity elements in $Z_{3,1} = \{00, 10, 20\}$ to each 2-tuple in S, where n-tuple addition is defined as in (3.1). Then, for any 2-tuple in S reducing the first digit modulo $k_1 = 3$ and the second digit modulo $k_2 = 1$ yields 00. Similarly, applying the same operations to any 2-tuple in C_1 yields 10 and to any 2-tuple in C_2 yields 20. Thus, any initial block comprising one treatment from the subgroup S and one from each of its cosets C_1 and C_2 will contain a complete representation of the 2-tuples in $Z_{3,1}$ after reducing modulo 3 on the first digit and modulo 1 on the second. This is a straightforward extension of David's criterion for a resolvable cyclic design.

One choice of initial block is (00 11 52). Cyclic development of this initial block yields the design shown in Table 5.4. It consists of six sets of four blocks with the first digit of each 2-tuple fixed within blocks.

Table 5.4: 2-Cyclic design for factors at $v_1 = 6$ and $v_2 = 4$ levels

00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33
11	12	13	10	21	22	23	20	31	32	33	30	41	42	43	40
52	53	50	51	02	03	00	01	12	13	10	11	22	23	20	21

40	41	42	43	50	51	52	53								
51	52	53	50	01	02	03	00								
32	33	30	31	42	43	40	41								

The resolvable layout of the design in Table 5.4 is obtained by grouping together the first and fourth, second and fifth, and third and last sets of four blocks, as shown in Table 5.5. Notice again that the first replicate in the resolvable layout can be obtained by adding each element in S in turn to every element in the initial block. Similarly, the second and third replicates are obtained by adding the elements of C_1

and C_2, respectively, to the initial block.

Table 5.5: Resolvable arrangement of the 2-cyclic design in Table 5.4

Replicate 1	Replicate 2	Replicate 3
00 01 02 03 30 31 32 33	10 11 12 13 40 41 42 43	20 21 22 23 50 51 52 53
11 12 13 10 41 42 43 40	21 22 23 20 51 52 53 50	31 32 33 30 01 02 03 00
52 53 50 51 22 23 20 21	02 03 00 01 32 33 30 31	12 13 10 11 42 43 40 41

In general, suppose that the number of treatments, v, and the block size, k, can be factorised as $v = v_1 v_2 \dots v_n$ and $k = k_1 k_2 \dots k_n$ such that $s_i = v_i/k_i$ where $s_i \in \mathbb{Z}^+$ $(i = 1, 2, \dots, n)$. Corresponding to the ith factor is the subgroup $< k_i >$, generated by k_i, of order s_i. Taking the direct sum over all n subgroups yields the set

$$S = \{a_1 a_2 \dots a_n : a_i \in < k_i >\}$$

It follows that S is a subgroup of $Z_{v_1,v_2,\dots,v_n}$ closed under n-tuple addition. Now, taking the direct sum over the n groups Z_{k_i} yields the set

$$Z_{k_1,k_2,\dots,k_n} = \{b_1 b_2 \dots b_n : b_i \in Z_{k_i}\}$$

The jth coset of S is obtained by adding the jth lexicographically ordered element in the set $Z_{k_1,k_2,\dots,k_n}$ to each element in S $(j = 1, 2, \dots, k-1)$, i.e.

$$C_j = \{d_1 d_2 \dots d_n : d_i = a_i + b_i \text{ modulo } v_i \text{ where } a_i \in < k_i > \text{ and } b_i \in Z_{k_i}\}$$

For example, consider again an experiment with $v = 24$ treatments with v factorised as $v_1 = 6$ and $v_2 = 4$. Now, however, the treatments are to be arranged in resolvable blocks of size $k = 4$ so that one possible factorisation of the block size satisfying $k = 4$ is $k_1 = k_2 = 2$. The subgroups generated by k_1 and k_2 are $\{0, 2, 4\}$ and $\{0, 2\}$, respectively, so that $S = \{00, 02, 20, 22, 40, 42\}$. The set $Z_{2,2} = \{00, 01, 10, 11\}$, is obtained by taking the direct sum of the groups Z_{k_1} and Z_{k_2} which, in this case, are both given by $\{0, 1\}$. With the exception of the identity element 00, addition of the jth ordered 2-tuple in $Z_{2,2}$ to each element in S yields its three cosets $C_1 = \{01, 03, 21, 23, 41, 43\}$, $C_2 = \{10, 12, 30, 32, 50, 52\}$ and $C_3 = \{11, 13, 31, 33, 51, 53\}$ $(j = 2, 3, 4)$. Cyclic development of any initial block

with one element from S and one from each of C_1, C_2 and C_3 will yield a resolvable 2-cyclic design. For example, one such initial block is (00 21 32 53), giving the 2-cyclic design with $r = 4$ distinct replicates shown in Table 5.6.

Table 5.6: Resolvable 2-cyclic design generated from the initial block (00 21 32 53).

Replicate 1	Replicate 2	Replicate 3	Replicate 4
00 02 20 22 40 42	01 03 21 23 41 43	10 12 30 32 50 52	11 13 31 33 51 53
21 23 41 43 01 03	22 20 42 40 02 00	31 33 51 53 11 13	32 30 52 50 12 10
32 30 52 50 12 10	33 31 53 51 13 11	42 40 02 00 22 20	43 41 03 01 23 21
53 51 13 11 33 31	50 52 10 12 30 32	03 01 23 21 43 41	00 02 20 22 40 42

As with cyclic designs, when v and k are relatively prime partial resolvable n-cyclic sets can be constructed. For instance, suppose that $v = v_1 v_2 \ldots v_n$ and $k = k_1 k_2 \ldots k_n$ and suppose that there exists some integer d^* that divides both v and k, i.e. $v/d^* \in \mathbb{Z}^+$ and $k/d^* \in \mathbb{Z}^+$. Further suppose that $d^* = d_1^* d_2^* \ldots d_n^*$ such that $p_i = v_i/d_i^*$ for $i = 1, 2, \ldots, n$. Then, $S^* = \{m_1 m_2 \ldots m_n : m_i \in < p_i >\}$ is a subgroup of order d^* of the group $Z_{v_1, v_2, \ldots, v_n}$. It is obtained by taking the direct sum over all subgroups $< p_i >$ of Z_{v_i} of order d_i^*. The cosets C_j^* (of S^*) are obtained by adding each of the elements in $Z_{p_1, p_2, \ldots, p_n}$ in turn to each of the elements in S^*. An initial block comprising one element from S and one from each of its $k - 1$ cosets will generate a resolvable n-cyclic design. Further, if the initial block comprises all the elements of S^* and all the treatment from $k/d^* - 1$ of its cosets, the resolvable n-cyclic design with $r = k/d^*$ distinct replicates will be generated.

For example, consider again the experiment with $v = 24$ treatments arranged in blocks of size $k = 4$. The initial block (00 21 30 51) comprises one treatment from the subgroup S and one treatment from each of its cosets C_1, C_2 and C_3. Cyclic development of this block will generate a resolvable 2-cyclic design. Now, let $d^* = 2$ so that d^* divides both v and k. If $v_1 = 6$, $v_2 = 4$ and $k_1 = k_2 = 2$ then one possible factorisation of d^* is $d_1^* = 2$ and $d_2^* = 1$. Hence, $p_1 = 3$ and $p_2 = 4$. It follows that $S^* = \{00, 30\}$. The initial block (00 21 30 51) is made up of all the treatments in S^* and all the treatments in its coset $C_9^* = \{21, 51\}$. Addition of each of the elements from S, C_1, C_2 and C_3 yields only $r = 2$ distinct replicates, as shown in Table 5.7.

The first replicate is generated by adding each of the elements from either S or C_2 to the initial block, and the second replicate by adding each of the elements from either C_1 or C_3. Note that were d^* factorised as $d_1^* = 1$ and $d_2^* = 2$, it would yield the subgroup $S^* = \{00, 02\}$. Then, any initial block would contain two treatments from S so that a partial resolvable 2-cyclic design could not be constructed.

Table 5.7: Partial resolvable 2-cyclic design generated from (00 21 30 51).

Replicate 1	Replicate 2
00 02 20 22 40 42	01 03 21 23 41 43
21 23 41 43 01 03	22 20 42 40 02 00
30 32 50 52 10 12	31 33 51 53 11 13
51 53 11 13 31 33	52 50 12 10 32 30

These resolvable n-cyclic designs belong to the family of α_n-designs. The columns of the r replicates are constructed from their respective first columns by cyclic substitution. The generating array α_n is obtained from these r initial columns. First, the elements are arranged into subgroup-coset order (to satisfy the α_n-design method of construction). They are then substituted with elements from $Z_{s_1,s_2,\ldots,s_n}$. Each n-tuple is relabelled according to its ordered position within its corresponding subgroup or coset; that is, if the n-tuple is the jth ordered element in its subgroup or coset, then it is substituted with the jth ordered element in $Z_{s_1,s_2,\ldots,s_n}$ $(j = 1, 2, \ldots, s)$.

For example, consider the design in Table 5.5. The four initial blocks of each replicate, with elements in subgroup-coset order, are (00 21 32 53), (22 01 50 53), (42 03 10 31) and (00 43 32 11) respectively. The generating array α_2 is obtained by substituting the 2-tuples in these blocks with the same ordered 2-tuple in the set $Z_{3,2} = \{00, 01, 10, 11, 20, 21\}$. For example, the 2-tuples in the first of these blocks, namely (00 21 32 53), are the first, third, fourth and sixth ordered 2-tuples in S, C_1, C_2 and C_3 respectively, and are therefore substituted with 00, 10, 11 and 21 respectively. The α_2-array for this design is shown in Table 5.8.

Table 5.8: Generating array corresponding to the design in Table 5.6

α_2-array			
00	00	00	00
10	21	10	21
11	11	11	11
21	00	21	00

5.4 Properties of n-cyclic designs

The properties of n-cyclic designs are now considered in the context of factorial experiments. John (1973) showed that n-cyclic designs have orthogonal factorial structure. This means that the canonical efficiency factors of the design can be directly associated with the main effects and interactions, and that appropriate optimality criteria for choosing between designs can be readily obtained.

The canonical efficiency factors are the non-zero eigenvalues of $(1/r)A$, where A is the information matrix of the design. The average efficiency factor E_x of a generalized interaction F^x is obtained from the harmonic mean of these canonical efficiency factors; see John and Williams (1995), (pp. 28-31, 202-204). For n-cyclic designs the canonical efficiency factors are given by

$$e_{u_1u_2\ldots u_n} = \frac{1}{r}\sum_{h_1=0}^{v_1-1}\sum_{h_2=0}^{v_2-1}\cdots\sum_{h_n=0}^{v_n-1}\xi_{h_1h_2\ldots h_n}\cos\Big\{\sum_{j=1}^{n}(2\pi u_j h_j/v_j)\Big\} \tag{5.1}$$

where $\xi_{h_1h_2\ldots h_n}$ are the elements from the first row of the information matrix, and where $u_l = 0$ for all l is excluded since $e_{00\ldots0} = 0$ ($u_l = 0, 1, \ldots, v_l - 1$; $l = 1, 2, \ldots, n$). The canonical efficiency factors associated with the main effects and interactions of a set of $s < n$ factors are obtained from (5.1) by letting $u_l = 0$ for the remaining $n - s$ factors. It can also be shown that they are given by the canonical efficiency factors of the s-cyclic design obtained from the full n-cyclic design by first deleting from each treatment combination the other $n - s$ factors and then deleting duplicate blocks.

As an example consider the resolvable design given in Table 5.5. The average efficiency factor of the main effect of factor F_1 is $E_{10} = 0.7435$ and is obtained from the harmonic mean of e_{u_10} for $u_1 = 1, 2, \ldots, 5$. The average efficiency factors

of the F_2 main effect and F_1F_2 interaction are $E_{01} = 0.8889$ and $E_{11} = 0.4715$ respectively. Other resolvable designs could be obtained by choosing different initial blocks, subject to the subgroup requirements of the previous section. For instance, the initial block (00 42 53) gives a resolvable design with E_{10} and E_{01} unchanged but $E_{11} = 0$, showing that some components of the interaction are not estimable. Another choice is the initial block (00 41 23) but now the main effect of the first factor is not estimable, as the cyclic design (0 4 2) is clearly disconnected.

Note that the design given in Table 5.5 is not the best 2-cyclic design of this size in terms of the main effects. The design generated from the initial block (00 12 33) has $E_{10} = 0.7843$, $E_{01} = 0.8889$ and $E_{11} = 0.4912$. However, it cannot be set out as a resolvable design since the initial block has two elements from the subgroup $S = \{00, 01, 02, 03, 30, 31, 32, 33\}$. As in the case of single factor experiments resolvability can involve a cost in terms of the optimality of the design.

One of the best resolvable designs, in terms of main effect average efficiency factors, obtained from a number of runs of the algorithm of Williams and John (1996) had $E_{10} = 0.7299$, $E_{01} = 0.8489$ and $E_{11} = 0.5296$. It is not as good as the resolvable 2-cyclic design in Table 5.5 on the main effects but has a higher average efficiency factor for the interaction.

5.5 Reduced resolvable n-cyclic designs

The resolvable 2-cyclic design given in Table 5.5 has three replicates. Suppose, however, that a design with only two replicates is required. A simple way to obtain such a design is to delete one of the replicates in Table 5.5. In general, reduced resolvable n-cyclic designs are obtained by deleting one or more replicates from a resolvable n-cyclic design.

First, the question of which replicates to delete will be considered. Let D be the resolvable n-cyclic design and let $D_{ij\ldots}$ be the design obtained by deleting replicates $i, j, \ldots$ from D. Two designs are isomorphic if one can be obtained from the other by a relabelling or permutation of the treatment labels.

Consider the design in Table 5.5. It can be seen that replicate 2 is obtained from

replicate 1, and replicate 3 from replicate 2, by adding 1, modulo 6, to the levels of the first factor. It follows that D_1, D_2 and D_3 are isomorphic to each other, i.e. $D_1 \sim D_2 \sim D_3$.

More generally, in any resolvable n-cyclic design, a replicate is obtained from the previous replicate by a similar simple permutation. Suppose that a resolvable n-cyclic design with $r = 4$ replicates is constructed and that a 2-replicate design is required. Now under this simple permutation it is clear that there are two equivalence groups of designs given by $D_{12} \sim D_{23} \sim D_{34} \sim D_{14}$ and $D_{13} \sim D_{24}$. Thus we have only two distinct designs, namely D_{12} and D_{13}. Dropping three replicates from a 6-replicate n-cyclic design, twenty designs are possible. These fall into four equivalence groups, generated from D_{123}, D_{124}, D_{125} and D_{135}, giving four distinct designs. The extension to the general case is straightforward.

The reduced designs are no longer n-cyclic designs; they are simply α_n-designs. Hence, they do not have orthogonal factorial structure. However, since they are α_n-designs the average efficiency factor given in (4.31) can be used to assess competing reduced n-cyclic designs.

Two examples of reduced designs obtained by deleting replicates from a full n-cyclic design are given below.

Example 5.1 *Table 5.9 gives resolvable designs for two factors at 4 and 3 levels in blocks of 4 based on the 2-cyclic design with initial block* (00 11 23 31). *For a design with $r = 3$ any one replicate can be dropped. For a design with $r = 2$ there are two possibilities and the choice between them is a trade-off between the second main effect and the interaction.*

Table 5.9: Resolvable block designs for factors at 4 and 3 levels.

r	Replicate deleted	E_{10}	E_{01}	E_{11}
4		1.0	0.9375	0.6319
3	1	1.0	0.9275	0.5994
2	1,2	1.0	0.9231	0.4800
2	1,3	1.0	0.8571	0.5882

Example 5.2 *Consider the factorial experiment to study the effect of four different pre-treatments on the germination of six Acacia mangium seedlots discussed in section 1.1. A three replicate design was required, with the replicates corresponding to the top, middle and bottom trays of the germination cabinet. It will now be considered how this experiment can be set out in blocks of size 6 with the primary aim being to maximise the average efficiency factors of the two main effects.*

Table 5.10: Resolvable block designs for $v_1 = 6$, $v_2 = 4$, $k = 6$ and $r = 3$.

Replicate deleted	E_{10}	E_{01}	E_{11}
1,2,3	1.0	0.9555	0.7451
1,2,4	1.0	0.9513	0.7492
1,2,5	1.0	0.9513	0.7492
1,3,5	1.0	0.9474	0.7461
Partial design	1.0	0.8889	0.5672
Best α_2-design	1.0	0.9600	0.7481
Williams-John	1.0	0.9489	0.7652

A resolvable partial 2-cyclic block design for two factors at $v_1 = 6$ and $v_2 = 4$ levels respectively, with $k = 6$ and $r = 3$ can be constructed. The design generated from the initial block (00 12 21 30 41 52) has $E_{10} = 1.0$, $E_{01} = 0.8889$ and $E_{11} = 0.5672$. Other initial blocks can be chosen, but they have the same efficiency factors.

The alternative is to construct a full resolvable 2-cyclic design with $r = k = 6$ and then delete three replicates. The initial block (00 12 21 30 41 53) gives a 6-replicate design with $E_{10} = 1.0$, $E_{01} = 0.9623$ and $E_{11} = 0.7928$; the best design in terms of first maximising E_{10} and E_{01}, and then maximising E_{11}. Table 5.10 gives the average efficiency factors of designs obtained by deleting three replicates from this design. Also given are the values for the partial (resolvable 2-cyclic) design and the best design obtained from running the Williams and John (1996) algorithm a number of times.

The best α_2-design in Table 5.10 is obtained from the generating array, α_2, in Table 5.11. This is best in terms of the main effects, although there is some loss in the interaction average efficiency factor compared with the design generated by

Table 5.11: Generating array, α_2, for $v_1 = 6$, $v_2 = 4$, $k = 6$ and $r = 3$

α_2-array		
01	01	10
10	11	00
11	00	11
01	10	00
10	11	11
00	00	01

the Williams and John algorithm. The partial 2-cyclic design does poorly on both E_{01} and E_{11}, as expected. Note that choosing the best α_2-design over the partial 2-cyclic design is at the expense of orthogonal factorial structure.

Chapter 6

Future directions

6.1 Thesis summary

This thesis has introduced a flexible family of resolvable incomplete block designs for factorial experiments, called α_n-designs. While the primary motivation for their development was their use in factorial experiments, it has been shown that they also improve on the available α-designs for some combinations of the design parameters v, k and r.

Two methods for the construction of α_n-designs were presented. First it was shown that the construction of α_n-designs is based on a straightforward extension of the method used by Patterson and Williams (1976) to construct α-designs. This method was primarily used in the thesis. An alternative method followed an approach given by Jarrett and Hall (1978).

Considerable attention has been devoted to studying the properties of α_n-designs. A particularly useful feature of these designs is the B^nC-structure of the treatment concurrence and information matrices. This property was exploited in deriving an expression for the average efficiency factor for the unstructured treatments, E, and for any generalized interaction in the factorial treatments, E_x. It reduced the problem of inverting a $v \times v$ matrix to that of inverting a small number of $k \times k$ matrices; see sections 3.7 and 4.4. In each case, the exploitation of the B^nC-structure led to substantial reductions in the number of calculations required. This feature translates into an algorithm which quickly finds the best available designs for the

given set of design parameters.

The search for a family of resolvable incomplete designs for factorial experiments has uncovered an important relationship between the family of n-cyclic designs and the family of α_n-designs, namely that *resolvable* n-cyclic designs are α_n-designs. Hence, although α_n-designs do not in general admit orthogonal factorial structure (OFS), a well-defined subset of them do; see sections 1.4 and 5.4.

The study of the relationship between resolvable n-cyclic and α_n-designs is incomplete. For instance, it is yet to be established how the $k \times r$ generating array α_n should be constructed so as to ensure that a resolvable n-cyclic design is obtained. Then, only a single design construction algorithm for both families of designs would be required. A direct consequence of this would be that the average efficiency factor of each main effect and interaction could be obtained from computationally cheaper methods appropriate for n-cyclic designs, instead of the method discussed in section 4.5 for α_n-designs.

A further question of interest is whether, for designs with r a multiple of k, the best α_n-designs are contained in the set of resolvable n-cyclic designs. If this were shown to be true in general, the choice of design would be clear. However, if it were not true, then it would be necessary to consider whether the gains in efficiency from using the best α_n-designs are worth sacrificing for less efficient resolvable n-cyclic designs with OFS. In this case it would be useful to have an objective measure of the cost of such a compromise. These issues are considered in the next section.

6.2 Search for efficient designs

The search for efficient designs requires a combination of theory and computing. The general methodology for the construction of α_n-designs has been defined, and writing a computer algorithm to do this is straightforward. However, for a given set of design parameters, numerous candidate designs exist. While the theory for assessing these designs has been developed in the sense that a closed form mathematical expression for calculating the average efficiency factor for any generalized interaction has been derived, the best way of using these criteria for choosing between different designs

is not entirely clear.

One approach is to use a single objective function based on a weighted average of the average efficiency factors of all factorial effects. An alternative approach is to use a sequential search. If, as is often the case in practice, main effects and two-factor interactions are more important than higher order interactions, an algorithm might first search for those α_n-designs which are optimal on the basis of main effects. At the next stage it would search for those designs that are optimal on the two-factor interactions, while keeping the main effects unchanged. In effect, it first finds the best allocation of each factor separately in the n-tuple treatment combination, and then finds how best to set out these factors in pairs. If required, this approach could be extended to second-order interactions and higher.

6.2.1 Objective functions

For single factor experiments one objective function is computed for each competing design, namely the average efficiency factor. Competing designs are then assessed by comparing their respective average efficiency factors. However, for factorial experiments there is an average efficiency factor for each main effect and interaction. The assessment of competing designs is no longer a straightforward process of comparing a single criterion. Rather, several criteria must be considered simultaneously for each design. How this should be done needs to be explored.

A simple and perhaps somewhat naive choice of objective function is to give all effects equal weight. In this case the objective function would be obtained by summing over the efficiency factors of all generalized interactions. However, such a choice assumes that all effects are of equal importance. This is seldom the case. For example, it is common in experiments with many factors for the high-order interactions (usually $n \geq 4$) to have a negligible effect on the response. Further, their physical meaning is often impossible to interpret. In such cases it would then be better to be able to estimate the lower order interactions with much greater precision than the higher order interactions.

An alternative is an objective function based on a weighted average of the average

efficiency factors of all main effects and interactions, i.e.

$$O = \sum_x w_x E_x$$

where w_x is the weight given to the average efficiency factor E_x of the generalized interaction F^x and $x \in \{x_1 x_2 \ldots x_n : x_i \in \{0,1\} \text{ and } \sum_i x_i \neq 0\}$. The summation $\sum_x$ is over all combinations of x. This objective function allows the experimenter to select the weights according to the known, or perceived, importance of each effect.

Optimisation algorithms for generating designs using a single objective function have been developed; see, for example, John and Whitaker (1993). Such algorithms are used extensively in the software package CycDesigN (Whitaker, Williams and John, 1997).

Recently, Eccleston and Whitaker (1999) proposed a method based on optimising multiple objective functions simultaneously. They applied their method to the problem of constructing optimal change-over designs. However, the approach could be applied to the construction of factorial designs. This is an interesting possibility, and will be the subject of further research.

6.2.2 Upper bounds

Theoretical upper bounds for the average efficiency factors are useful to judge how good a design is, and whether there is any possible scope for further improvement. For instance, if a search algorithm produces a design whose important average efficiency factors are within 1%, say, of these theoretical bounds then the search algorithm can be safely terminated.

There have been numerous contributions to the literature on upper bounds for the average efficiency factor in single factor experiments. These can be used to obtain bounds for main effects. In comparison, the available literature on upper bounds for factorial average efficiency factors is quite modest. Dean and Lewis (1983) showed that for any two-factor experiment arranged as an incomplete block design, and which possesses orthogonal factorial structure, the interaction average efficiency factor, E_{11}, can be written as a function of the average efficiency factor of the unstructured treatments, E, and the average efficiency factors of the main

effects, E_{10} and E_{01}, i.e.

$$(v_1 - 1)(v_2 - 1)E_{11}^{-1} = (v - 1)E^{-1} - (v_1 - 1)E_{10}^{-1} - (v_2 - 1)E_{01}^{-1}$$

It follows that for such designs an upper bound for E_{11} must take into account the average efficiency factors of the main effects. Dean and Lewis first derived upper bounds for E_{11} for designs in which one or both main effects would be estimated with full precision. They later extended these results to the case where neither main effect achieves full efficiency, and also generalized the bound to higher order interactions; see Lewis and Dean (1984). These results might provide a useful starting point in developing upper bounds for resolvable n-cyclic designs. However, they are unsuitable for α_n-designs which do not possess OFS.

Another possibility is to obtain the upper bounds recursively, following the relation for the average efficiency factor of a generalized interaction, F^x, given in (4.35), i.e.

$$\nu_x U_x^{-1} = U_{T_x} - \sum_{y_x} \nu_y U_y^{-1}$$

where U_x and U_{T_x} are upper bounds for E_x and T_x respectively.

Much work is needed in the area of upper bounds for factorial experiments. In particular, upper bounds for α_n-designs should take into account the resolvable structure of these designs since it is expected that incorporating this structural constraint will yield tighter bounds.

6.2.3 Algorithm for the generation of α_n-designs

A program for the generation of α_n-designs is presented in Appendix D. It is based on a program in CycDesigN (Whitaker, Williams and John, 1997) used to generate α-designs, and is written in the C/C++ programming language.

The program is an exchange algorithm. The user must initially specify the parameter values for the required design. First, a randomly chosen $k \times r$ generating array whose elements are selected from the set Z_s is constructed. An element, chosen at random, is then exchanged with another randomly chosen candidate element from Z_s. A decision is then made to accept or reject this design. The acceptance

rule is based on the nested simulated annealing algorithm discussed in John and Whitaker (1993). Designs which improve the value of the objective function are always accepted. However, an exchange that leads to a worse value for the objective function can also be accepted with a small probability. This feature of the simulated annealing algorithm is desirable because it overcomes local optima problems.

The program uses an objective function based on a weighted average of the efficiency factors of all factorial effects up to and including three-way interactions. The weights are specified by the user.

The user is able to intervene and stop the program manually. This should be done after running the algorithm for a period of time, or when the average efficiency factors of the design are reasonably close to their theoretical upper bounds. Currently, the program simply uses the sum of the weights as an upper bound for the objective function. It is recognised that incomplete block designs with $k < v$ cannot reach this upper bound and so the program must be terminated manually.

A flowchart summarising the main processes used by the program is presented in Figure 6.1.

While the program generates efficient designs very quickly, evaluating E_x for each design is very time consuming. Screening functions such as those based on the (M, S)-optimality criterion could be used to reduce the number of calculations. However, recent work by John and Whitaker (2000) and Williams and John (2000) on recursive methods for updating the average efficiency factors has shown that algorithms using this method consistently produce more efficient designs faster. This is an area for future research.

6.3 Latinization

Another blocking feature used frequently in practice is Latinization. This is where the replicates are set out under each other. In this case the design can also be viewed as a resolvable block design with s long blocks of size rk, where $s = v/k$. Now it is desirable that no treatment combination occurs more than once in any long block. For a discussion of Latinized designs see Williams (1986). Resolvable n-cyclic

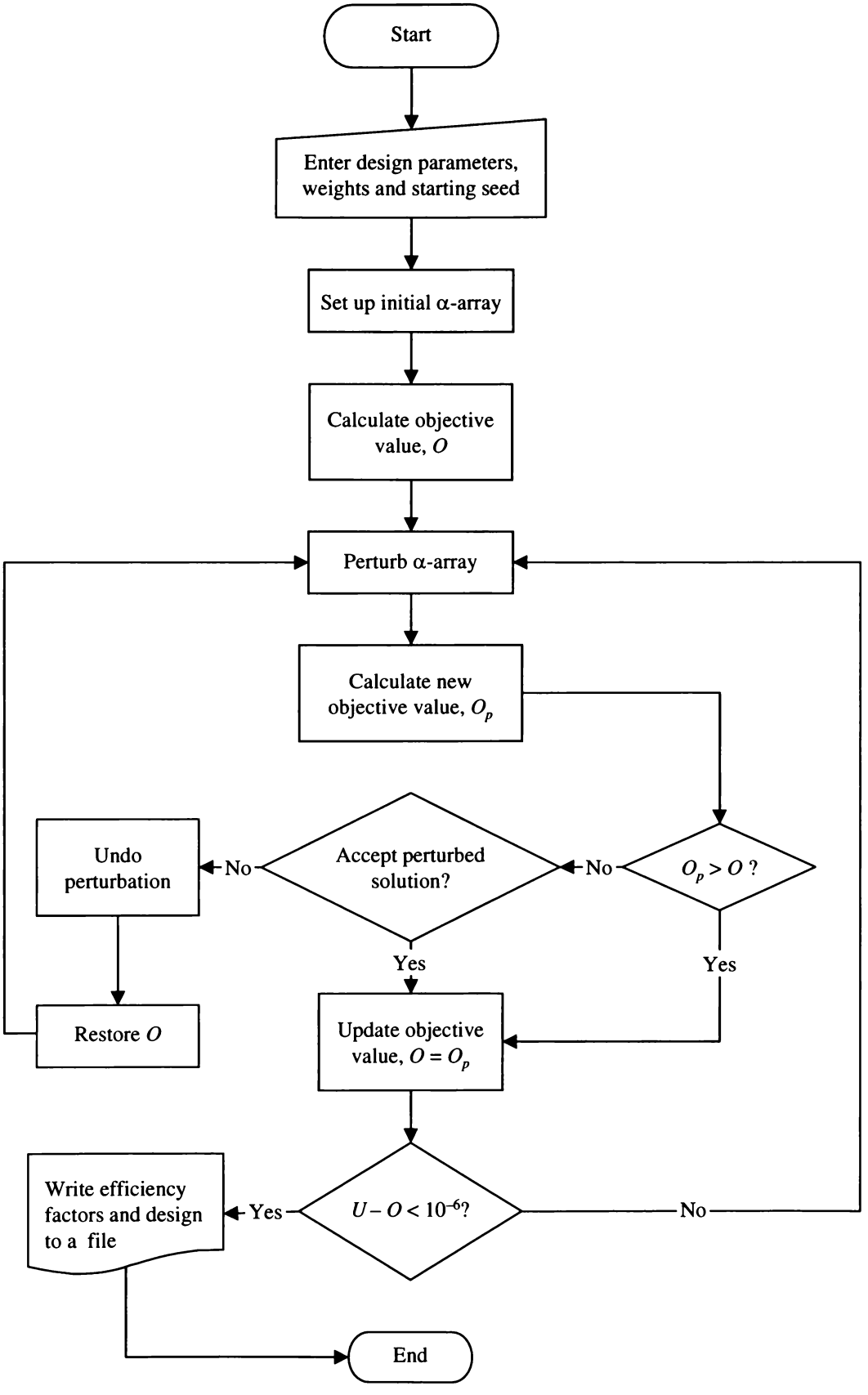

Figure 6.1: Flowchart of a program for the generation of α_n-designs

designs, and the reduced designs derived from them, are automatically Latinized because of the cyclic method of construction. For instance, if the three replicates of the design in Table 5.5 were contiguous no treatment combination would occur more than once in any of the eight long blocks of size 9.

Latinized designs with $r \leq s$ can also be constructed for factorial experiments using α_n-designs. Restrictions must be placed on the elements $\alpha_n(i,j)$ of the generating array to prevent the same factorial treatment from occurring more than once in any long block ($i = 1, 2, \ldots, k;\ j = 1, 2, \ldots, r$). An α_n-design will have the Latinized property in the unstructured treatments if

$$\alpha_n(i,j) - \alpha_n(i,j') \neq 00\ldots0$$

for $j \neq j'$, where n-tuple subtraction is defined as in (3.9) and (3.10).

Latinization in the unstructured treatments does not guarantee that a design will have good properties in terms of the efficiencies of the main effects. For example, consider the Latinized α_2-design for $v_1 = 6$, $v_2 = 4$, $k = 6$ and $s = 6$ shown in Table 6.1. Note that level 0 of factor F_1 occurs three times in the first and last two long blocks of the design, and not at all in the middle two. One way of ensuring that factor levels are spread evenly across long blocks would be to use an objective function for the long block structure as well as the short blocks. This is another situation in which the simultaneous optimisation of a pair of objective functions, following the method proposed by Eccleston and Whitaker (1999), could be fruitful and needs investigating.

6.4 Further work

The discovery and subsequent development of α_n-designs has exposed a new set of interesting questions which must be followed-up. Some of these have already been discussed, namely

- the need for further study of the relationship between α_n-designs and resolvable n-cyclic designs,

Table 6.1: Latinized α_2-design for $v_1 = 6$, $v_2 = 4$, $k = 4$ and $s = 6$

Replicate 1	00	01	10	11	20	21
	13	12	23	22	03	02
	31	30	41	40	51	50
	52	53	32	33	42	43
Replicate 2	10	11	20	21	00	01
	02	03	12	13	22	23
	40	51	50	31	30	41
	52	33	32	43	42	53
Replicate 3	10	11	20	21	00	01
	03	02	13	12	23	22
	31	30	41	40	51	50
	42	43	52	53	32	33

- the simultaneous optimisation of multiple objective functions in the search for efficient α_n-designs,
- the development of upper bounds for factorial efficiency factors which take into account the resolvable structure of α_n-designs,
- the incorporation of the recursive methods of John and Whitaker (2000) and Williams and John (2000) in a program for the generation of α_n-designs, and
- the use of α_n-designs in the construction of Latinized designs for factorial experiments.

The class of α-designs have been found to be useful in a wide range of experimental situations that arise in practice. For instance, α-designs have been used for the construction of resolvable row-column designs, i.e. experiments in which variation is controlled in two dimensions. They have also been used in experiments involving nested treatment structures; for example, forestry trials in which a number of seedlots are collected from several regions so that seedlots are specific to, or nested within regions. In fact, John and Williams (1999) have considered the use of α-designs for the construction of resolvable row-column designs when the treatments have a nested structure.

Since α_n-designs are an extension of α-designs, it seems plausible that this application to a wide range of experimental situations should extend to α_n-designs. Certainly there are situations in practice where it is desirable to construct resolvable row-column designs for factorial experiments. For instance, consider a forestry trial in which it is planned to determine the best combination of seedlots and irrigation methods for a given response variable. If the experiment were set out in a glasshouse, a row-column design would be appropriate for controlling for possible variation in conditions (within the glasshouse) in two dimensions. While α_n-designs are in themselves poor row-column designs because the row-component design is disconnected, they might prove to be useful in constructing efficient row-column designs if they are constructed in two steps. The first step would be to find the optimal column-component design. The second step would require rearranging the treatments within columns using an interchange algorithm such as simulated annealing to find an efficient row-component design.

Appendix A

Some group theory definitions and results

A *group* is a non-empty set G, together with a binary operation $(\cdot)$ on G, which satisfies the following three properties.

i) The operation is *associative*; that is, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

ii) There is an *identity* element e in G such that $e \cdot a = a \cdot e = a$ for all $a \in G$.

iii) For each element a in G, there is an element b in G such that $a \cdot b = b \cdot a = e$.

A consequence of these three properties is that any pair of elements a and b in G can be combined without going outside the set; that is, G is *closed* under the group operation. A group G is *abelian* if for every pair of elements a and b in G, $a \cdot b = b \cdot a$; that is, if the group operation is commutative.

Let H be a subset of the group G. If H is itself closed under the binary operation of G, then H is called a *subgroup* of G.

For any a in G, the set $\{h_i + a : h_i \in H\}$ is denoted by H_a and is called the *left coset of H in G*. Similarly, ${}_aH = \{a + h_i : h_i \in H\}$ is the *right coset of H in G*. When the group G is abelian, $H_a = {}_aH$.

Let $< a >$ denote the set $\{na : n \in \mathbb{Z}\}$. Further, let a be any element of the group G. Then $< a >$ is the *cyclic subgroup of G generated by a*. If $G =< a >$ then

a is a generator of G and hence, G is a *cyclic group*. Cyclic groups are *abelian*.

Now, let n be a fixed positive integer and let a and b be any integers. The number r such that

$$a + b = nq + r$$

for $0 \leq r < n$, is the *sum of a and b modulo n*. The finite set of integers $Z_v = \{0, 1, \ldots, v-1\}$ forms a cyclic group under addition modulo v.

Groups may be pieced together to form larger groups. Let $G_1, G_2, \ldots, G_n$ be a finite collection of groups and let $g_i \in G_i$ for $i = 1, 2, \ldots, n$. Then, the *direct sum* of $G_1, G_2, \ldots, G_n$, denoted as $G_1 \oplus G_2 \oplus \ldots \oplus G_n$, is the set of all n-tuples $g_1 g_2 \ldots g_n$ for which the ith component is an element of G_i. In symbols,

$$G_1 \oplus G_2 \oplus \ldots \oplus G_n = \{g_1 g_2 \ldots g_n : g_i \in G_i\}$$

The above definitions and results can be found in any standard text on abstract algebra; see, for example, Fraleigh (1982).

Appendix B

Matrix results

The following definitions and results were found useful in the study of α_n-designs. They fall into four main categories:

i) idempotent matrices; see Definition B.1 and Theorems B.1 and B.2,

ii) circulant matrices; see Definitions B.2 and B.3 and Theorems B.3 and B.4,

iii) Kronecker products; see Definition B.4 and Theorems B.5 and B.6, and

iv) Hermitian matrices; see Definitions B.6 and B.7 and Theorems B.8 – B.11.

Most of the definitions and theorems that are presented have been directly transcribed from the texts by Searle (1982), Graybill (1987) and Anton (1987). Proofs of all theorems, except Theorems B.9 – B.11, have been omitted since they can be found in the above mentioned texts.

Theorems B.9 – B.11 were used in sections 2.6 and 4.5 to show that complex matrix inversion could be avoided when calculating average efficiency factors for α-designs and α_n-designs respectively. It was necessary to derive these theorems and hence, their proofs are presented.

Definition B.1 *An $n \times n$ matrix A is idempotent if $A^2 = A$.*

Theorem B.1 *If the $n \times n$ matrix A is idempotent then rank(A) =trace(A). (See Searle, 1982, p.320)*

Theorem B.2 *If A is an idempotent matrix, then $I_n - A$ is idempotent with $rank(I_n - A) = n - rank(A)$. (See Graybill, 1987, p.439)*

Definition B.2 *A square matrix Γ_h $(h = 0, 1, \ldots, s-1)$ whose first row has one in the $(h+1)$th column and all other elements zero and elements in each successive row move to the right one position (with wrap-around at the end) is called a basic circulant.*

Theorem B.3 *Let Γ_h be an $s \times s$ basic circulant matrix $(h = 0, 1, \ldots, s-1)$. Let $\omega_0, \omega_1, \ldots, \omega_{s-1}$ be the sth roots of unity. Then, the normalised eigenvectors of Γ_h are given by*

$$\gamma_u = \frac{1}{\sqrt{s}} \begin{pmatrix} \omega_u^0 \\ \omega_u^1 \\ \omega_u^2 \\ \vdots \\ \omega_u^{s-1} \end{pmatrix}$$

with corresponding eigenvalues $\lambda_u = \omega_u^h$ for $u = 0, 1, \ldots, s-1$. (See Graybill, 1987, p.239)

Definition B.3 *A square matrix C whose first row equals $(c_0, c_1, \ldots, c_{s-1})$ and elements in each successive row move to the right one position (with wrap-around at the end) is called a regular circulant.*

Theorem B.4 *Let C be an $s \times s$ regular circulant with the first row equal to $(c_0, c_1, \ldots, c_{s-1})$. Let $\omega_0, \omega_1, \ldots, \omega_{s-1}$ be the s roots of unity. The eigenvalues of C are given by $\lambda_0, \lambda_1, \ldots, \lambda_{s-1}$, where for each $i = 1, 2, \ldots, s-1$*

$$\lambda_i = c_0\omega_i^0 + c_1\omega_i^1 + \cdots + c_{s-1}\omega_i^{s-1}$$

and $(x_0, x_1, \ldots, x_{s-1})$ are the corresponding eigenvectors, where

$$x_i = \begin{pmatrix} \omega_i^0 \\ \omega_i^1 \\ \vdots \\ \omega_i^{s-1} \end{pmatrix}$$

(See Graybill, 1987, p.239)

Definition B.4 *Let A be an $m \times n$ matrix and let B be a $p \times q$ matrix. The Kronecker product of A and B, denoted $A \otimes B$, is the $mp \times nq$ matrix defined as*

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}$$

Theorem B.5 *Let a be any scalar and A, B and C be any matrices; then*

i) $(aA) \otimes B = A \otimes (aB) = a(A \otimes B)$

ii) $(A \otimes B) \otimes C = A \otimes (B \otimes C)$

iii) $(A \otimes B)' = A' \otimes B'$

iv) $trace(A \otimes B) = trace(A)trace(B)$

(See Graybill, 1987, p.220-223)

Theorem B.6 *Let A_i be $m_1 \times n_1$ matrices and B_i be $m_2 \times n_2$ matrices for $i = 1, 2, \ldots, k$. Then*

$$\sum_{i=1}^{k} A_i \otimes \sum_{j=1}^{k} B_j = \sum_{i=1}^{k} \sum_{j=1}^{k} (A_i \otimes B_j)$$

(See Graybill, 1987, p.228)

Definition B.5 *The n-component vector space $\mathbb{R}^n$ is defined to be a Euclidean space, denoted by E_n, if and only if the distance between any two points (vectors) $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined to be $d = \left[\sum_{i=1}^{n} (a_i - b_i)^2\right]^{1/2}$ (See Graybill, 1987, p.53)*

Theorem B.7 *Let C be any $n \times n$ matrix. Then $x'Cx = 0$ for all $x \in E_n$ if and only if $C = -C'$, that is, if and only if C is a skew-symmetric matrix. (See Graybill, 1987, p.415)*

Definition B.6 *A square matrix H with complex entries is called* **Hermitian** *if $H = H^{\dagger}$, where $H^{\dagger}$ denotes the conjugate transpose of H. (See Anton, 1987, p.466)*

Definition B.7 *A square matrix having the property $Y = -Y'$ is called skew-symmetric. Its diagonal elements are zero and each off-diagonal element is minus its symmetric partner; i.e. $y_{ii} = 0$ and $y_{ij} = -y_{ji}$. (See Searle, 1982, p.65)*

Theorem B.8 *If X and Y are matrices with complex entries and c is any complex number then*

i) $(X^\dagger)^\dagger = X$

ii) $(X+Y)^\dagger = X^\dagger + Y^\dagger$

iii) $(cX)^\dagger = \bar{c}X^\dagger$

iv) $(XY)^\dagger = Y^\dagger X^\dagger$

(See Anton, 1987, p.464)

Theorem B.9 *Let A_h be a $k \times k$ matrix with real entries and let $A'_h = A_{s-h}$. Then, the matrix A^*_u defined as*

$$A^*_u = \sum_{h=0}^{s-1} \omega_u^h A_h \tag{B.1}$$

where

$$\omega_u = \exp(2\pi iu/s) = \cos(2\pi iu/s) + i\sin(2\pi iu/s) \tag{B.2}$$

is the uth of the sth roots of unity, is Hermitian ($u, h = 0, 1, \ldots, s-1$).

Proof. Let the matrix A^*_u be defined as in (B.1). Then A^*_u is a Hermitian matrix if $A^*_u = (A^*_u)^\dagger$, that is, if A^*_u equals its conjugate transpose.

From (B.1), (B.2) and Theorem B.8, and since A_h consists of only real entries, taking the conjugate transpose of A^*_u gives

$$(A^*_u)^\dagger = \sum_{h=0}^{s-1} \omega_u^{-h} A'_h$$

Now, since $\omega_u^{-h} = \omega_u^{s-h}$ and A_h is circulant, it follows that

$$\begin{aligned}(A^*_u)^\dagger &= \sum_{h=0}^{s-1} \omega_u^{-h} A_{s-h} \\ &= \sum_{h=0}^{s-1} \omega_u^h A_h \\ &= A^*_u\end{aligned}$$

Hence A^*_u is Hermitian.

End of proof.

Theorem B.10 *Let $A_{h_1h_2\ldots h_n}$ be a $k_j \times k_j$ matrix with real entries and let $A'_{h_1h_2\ldots h_n} = A_{g_1g_2\ldots g_n}$, where $h_j = s_j - h_j$. Then, the matrix $A^*_{u_1u_2\ldots u_n}$ defined as*

$$A^*_{u_1u_2\ldots u_n} = \sum_{h_1=0}^{s_1-1} \sum_{h_2=0}^{s_2-1} \cdots \sum_{h_n=0}^{s_n-1} \omega_{u_1}^{h_1} \omega_{u_2}^{h_2} \ldots \omega_{u_n}^{h_n} A_{h_1h_2\ldots h_n} \tag{B.3}$$

where

$$\omega_{u_j} = \exp(2\pi i u_j/s) = \cos(2\pi i u_j/s_j) + i \sin(2\pi i u_j/s_j) \tag{B.4}$$

is the u_jth root of unity, is Hermitian ($u_j, h_j = 0, 1, \ldots, s_j - 1;\ j = 1, 2, \ldots, n$).

Proof. This is a straightforward extension of the proof of Theorem B.9.

Let the matrix $A^*_{u_1u_2\ldots u_n}$ be defined as in (3.15). Then $A^*_{u_1u_2\ldots u_n}$ is a Hermitian matrix if $A^*_{u_1u_2\ldots u_n} = (A^*_{u_1u_2\ldots u_n})^\dagger$, that is, if $A^*_{u_1u_2\ldots u_n}$ equals its conjugate transpose.

Since the matrices $A_{h_1h_2\ldots h_n}$ in $A^*_{u_1u_2\ldots u_n}$ consist of only real entries and the u_jth roots of unity are defined in (B.4) are complex valued numbers, the conjugate transpose of $A^*_{u_1u_2\ldots u_n}$ is obtained by taking the complex conjugate of each $\omega_{u_j}^{h_j}$ and the transpose of the matrices $A_{h_1h_2\ldots h_n}$, i.e.

$$(A^*_{u_1u_2\ldots u_n})^\dagger = \sum_{h_1=0}^{s_1-1} \sum_{h_2=0}^{s_2-1} \cdots \sum_{h_n=0}^{s_n-1} \omega_{u_1}^{-h_1} \omega_{u_2}^{-h_2} \cdots \omega_{u_n}^{-h_n} A'_{h_1h_2\ldots h_n}$$

Since $\omega_{u_j}^{-h_j} = \omega_{u_j}^{s_j-h_j}$ and $A'_{h_1h_2\ldots h_n} = A_{g_1g_2\ldots g_n}$, it follows that

$$(A^*_{u_1u_2\ldots u_n})^\dagger = \sum_{g_1=0}^{s_1-1} \sum_{g_2=0}^{s_2-1} \cdots \sum_{g_n=0}^{s_n-1} \omega_{u_1}^{g_1} \omega_{u_2}^{g_2} \cdots \omega_{u_n}^{g_n} A_{g_1g_2\ldots g_n} = A^*_{u_1u_2\ldots u_n}$$

where $g_j = s_j - h_j$ $(j = 1, 2, \ldots, n)$. Hence, $A^*_{u_1u_2\ldots u_n}$ is a Hermitian matrix.

End of proof.

Theorem B.11 *The inverse of a Hermitian matrix is Hermitian, i.e. if H is a Hermitian matrix then H^{-1} is also a Hermitian matrix, i.e. if $H = H^\dagger$ then $H^{-1} = (H^{-1})^\dagger$.*

Proof. We want to show that if H is a Hermitian matrix then H^{-1} is also a Hermitian matrix, i.e. if $H = H^\dagger$ then $H^{-1} = (H^{-1})^\dagger$.

Let H be a Hermitian matrix and let H^{-1} be the inverse of H. Hence, $HH^{-1} = I$. Since H is Hermitian and using the results in Theorem B.8, taking the conjugate

transpose of both sides gives $(H^{-1})^{\dagger}H = I$. This result holds iff $(H^{-1})^{\dagger} = H^{-1}$. Hence, if H is a Hermitian matrix then its inverse, H^{-1}, is also Hermitian.

End of proof.

Appendix C

Publication

ELSEVIER

Journal of Statistical Planning and Inference 77 (1999) 293–299

journal of statistical planning and inference

Resolvable block designs for factorial experiments

J.A. John *, Katya Ruggiero

Department of Statistics, University of Waikato, Private Bag 3105, Hamilton, New Zealand

Received 17 December 1997; accepted 16 September 1998

Abstract

It is known that n-cyclic designs provide a flexible class of designs suitable for setting out factorial experiments. In this paper we show that many of these designs are resolvable. Further, an extensive class of practically useful designs can be derived from them by deleting replicates. The properties of the designs compare favourably with those obtained by the algorithm of Williams and John (1996) (Appl. Statist. 45, 39–46).

Keywords: α-designs; Average efficiency factor; Cyclic designs; Factorial structure; Interactions; Latinised designs; Main effects; Resolvable designs

1. Introduction

Resolvable block designs are widely used in the design of variety trials. Often these trials involve a number of treatment factors so that a factorial experiment is required. For example, seedlots might be subjected to different pre-germination treatments. Interest would centre on differences between seedlots and between pre-germination treatments, and on whether pre-germination treatment affects different seedlots in different ways. Lattice and α-designs, the commonly used resolvable designs, are not suitable for such experiments since they are usually constructed assuming that all treatment comparisons are of equal interest. In factorial experiments, however, it is often desirable to estimate some treatment comparisons with a greater precision than others; e.g. main effect comparisons at the expense of interactions.

Williams and John (1996) described a computer algorithm to construct resolvable block and row-column designs for factorial experiments. The algorithm sets out to ensure that each level of every treatment factor occurs as equally as possible in each block of the design, or in each row and column. The expectation then is that main effects, at least, are estimated with a high degree of precision. While the algorithm does often produce practically useful designs, it is not entirely satisfactory. The aim of the

* Corresponding author. Tel.: +64 7 838 4073; fax: +64 7 838 4155; e-mail: nyc@waikato.ac.nz.

algorithm is intuitively reasonable, but some of the generated designs have relatively low main effect precision. It is possible that a design will not allow a main effect to be estimated at all, i.e. the design is disconnected with respect to that main effect. Further, the precision with which interactions are estimated is not directly considered.

In this paper we shall examine the class of n-dimensional cyclic designs, called n-cyclic designs, known to be suitable as non-resolvable designs for factorial experiments. These designs possess the important property of factorial structure, and the precision with which main effects and interactions can be estimated is readily determined. In the next section we show that a sizeable subset of n-cyclic designs can be set out in replicates, and that they provide a useful class of resolvable designs for factorial experiments. The properties of the designs are examined in Section 3. There are however, constraints on the number of replicates. It may be that a design with fewer replicates than that given by the n-cyclic design will be required. In Section 4 we consider reducing the size of the design by deleting replicates from the full n-cyclic design. Although the attractive properties of the cyclic designs are now lost, the resulting designs are of practical use and compare favourably with the designs obtained from the algorithm of Williams and John (1996).

2. n-Cyclic designs

Cyclic designs are obtained by a cyclic development of one or more initial blocks; see John and Williams (1995), (Ch. 3) for a detailed discussion of cyclic designs. If each treatment label is represented by a set of n digits and the cyclic development is carried out on each digit in turn then the design is called an n-cyclic design. We shall let the n digits correspond to n treatment factors, and will let the ith factor F_i be at the m_i levels $0,1,\ldots,m_i-1$ $(i=1,\ldots,n)$. Consider the design given in Table 1 for $n=2$ factors in blocks of size $k=3$ with $m_1=6$ and $m_2=4$. Blocks are first generated from the initial block (00 11 52) by cycling each factor in turn. These blocks are then rearranged into replicates to give the resolvable block design in Table 1; where the blocks are written in columns.

Each replicate in Table 1 consists of two sets of four blocks. The four blocks of the second set can be obtained by adding 3, modulo 6, to the levels of the first factor in the first set. This means that the levels of the first factor in any block consists of one element from the subgroup $S=\{0,3\}$ and one element from each of the cosets of S, namely $C_1=\{1,4\}$ and $C_2=\{2,5\}$; levels of the second factor are immaterial.

Table 1
Resolvable block design for factors at 6 and 4 levels

Replicate 1	Replicate 2	Replicate 3
00 01 02 03 30 31 32 33	10 11 12 13 40 41 42 43	20 21 22 23 50 51 52 53
11 12 13 10 41 42 43 40	21 22 23 20 51 52 53 50	31 32 33 30 01 02 03 00
52 53 50 51 22 23 20 21	02 03 00 01 32 33 30 31	12 13 10 11 42 43 40 41

In general, suppose we factorise the block size as $k = k_1 k_2 \cdots k_n$ such that k_i divides m_i $(i = 1, \ldots, n)$. Corresponding to the ith factor is a subgroup $S_i = \{0, k_i, 2k_i, \ldots, (d_i - 1)k_i\}$ of the factor levels $0, 1, \ldots, m_i - 1$ of order $d_i = m_i/k_i$ $(i = 1, \ldots, n)$. Let $S = S_1 \times S_2 \times \cdots \times S_n$ be a concatenation of the elements of these subgroups, i.e. the elements of S comprise n-tuples of the form $a_1 a_2 \cdots a_n$, where $a_i \in S_i$ for all i. S is a subgroup of order $\prod_{i=1}^{n} d_i$ under addition

$$a_1 a_2 \cdots a_n + b_1 b_2 \cdots b_n = c_1 c_2 \cdots c_n \tag{1}$$

where $a_i + b_i = c_i$ modulo m_i $(i = 1, \ldots, n)$. Then if the initial block of the n-cyclic design consists of one element from S and one from each of its cosets C_j $(j = 1, \ldots, k - 1)$ the design is resolvable. The $r = k$ replicates are obtained by adding the elements of $S, C_1, \ldots, C_{k-1}$ in turn to the elements in the initial block, where addition is defined in Eq. (1).

For instance, consider designs for two factors at 6 and 4 levels in blocks of size 4 with $k_1 = 2$ and $k_2 = 2$. Then $S_1 = \{0, 2, 4\}$ and $S_2 = \{0, 2\}$ so that $S = \{00, 02, 20, 22, 40, 42\}$ with cosets $C_1 = \{01, 03, 21, 23, 41, 43\}$, $C_2 = \{10, 12, 30, 32, 50, 52\}$ and $C_3 = \{11, 13, 31, 33, 51, 53\}$. Any initial block with one element from S and one from each of C_1, C_2 and C_3 will give a resolvable 2-cyclic design. For example, one such initial block is (00 21 32 53), giving a design with $r = 4$ distinct replicates. For the design in Table 1, $k_1 = 3$ and $k_2 = 1$ and the subgroup S is a concatenation of the subgroups $S_1 = \{0, 3\}$ and $S_2 = \{0, 1, 2, 3\}$.

Partial resolvable n-cyclic designs with $r < k$ replicates can also be constructed. Suppose that the initial block of a resolvable n-cyclic design comprises a subgroup S^* of order d^* together with $(k/d^*) - 1$ of its cosets. Then the resulting resolvable n-cyclic design will have $r = k/d^*$ distinct replicates. For example, consider again the 6×4 experiment in blocks of size 4. Suppose the initial block is (00 21 30 51). This block will generate a resolvable 2-cyclic design since it comprises one element from S and one element from each of its cosets C_1, C_2 and C_3. Further, the initial block is made up of the subgroup $S^* = \{00, 30\}$ and its coset $\{21, 51\}$. Hence, the design has $r = 2$ distinct replicates. Note, however, that the second factor only occurs at levels 0 and 1 in the initial block, so that this main effect will be estimated relatively inefficiently.

3. Properties of n-cyclic designs

A design has factorial structure if any treatment contrast belonging to one factorial effect (main effect or interaction) can be estimated independently of any other effect. That is, the factorial effects are mutually orthogonal. This means that the canonical efficiency factors of the design can be directly associated with the main effects and interactions, and that appropriate optimality criteria for choosing between designs can be readily obtained. All n-cyclic designs have factorial structure.

The canonical efficiency factors are obtained from the non-zero eigenvalues of the information matrix of the design; see John and Williams (1995), (pp. 28–31, 202–204).

For n-cyclic designs the canonical efficiency factors are given by

$$e_{u_1,\ldots,u_n} = (1/r) \sum_{h_1=0}^{m_1-1} \cdots \sum_{h_n=0}^{m_n-1} \xi_{h_1,\ldots,h_n} \cos\left\{ \sum_{j=1}^{n} (2\pi u_j h_j / m_j) \right\}$$
$$(u_l = 0,\ldots,m_l - 1; l = 1,\ldots,n) \tag{2}$$

where $\xi_{h_1\ldots h_n}$ are elements from the first row of the information matrix, and where $u_l = 0$ for all l is excluded since $e_{00\ldots0} = 0$. The canonical efficiency factors associated with the main effects and interactions of a set of $s < n$ factors are obtained from Eq. (2) by letting $u_l = 0$ for the remaining $n - s$ factors. It can also be shown that they are given by the canonical efficiency factors of the s-cyclic design obtained from the full n-cyclic design by first deleting from each treatment combination the other $n - s$ factors and then deleting duplicate blocks. An average efficiency factor can be determined for any factorial effect as the harmonic mean of its corresponding canonical efficiency factors. These factors are used for comparing different designs.

As an example consider the resolvable design given in Table 1. The average efficiency factors of the main effects are $E_1 = 0.7435$ and $E_2 = 0.8889$, respectively, and the interaction is $E_{12} = 0.4715$. Other resolvable designs could be obtained by choosing different initial blocks, subject to the subgroup requirements of the previous section. For instance, the initial block (00 42 53) gives a resolvable design with E_1 and E_2 unchanged but $E_{12} = 0$, showing that some components of the interaction are not estimable. Another choice is the initial block (00 41 23) but now the main effect of the first factor is not estimable, as the cyclic design (0 4 2) is clearly disconnected.

Note that the design given in Table 1 is not the best 2-cyclic design of this size in terms of the main effects. The design generated from the initial block (00 12 33) has $E_1 = 0.7843$, $E_2 = 0.8889$ and $E_{12} = 0.4912$. However, it cannot be set out as a resolvable design since the initial block has two elements from the subgroup $S = \{00, 01, 02, 03, 30, 31, 32, 33\}$. As in the case of single factor experiments resolvability can involve a cost in terms of the optimality of the design.

One of the best resolvable designs, in terms of main effect average efficiency factors, obtained from a number of runs of the algorithm by Williams and John (1996) had $E_1 = 0.7299$, $E_2 = 0.8489$ and $E_{12} = 0.5296$. It is not as good as the 2-cyclic design on the main effects but has a higher average efficiency factor for the interaction.

4. Reduced resolvable n-cyclic designs

The resolvable 2-cyclic design given in Table 1 has three replicates. Suppose, however, that a design with only two replicates is required. A simply way to obtain such a design is to delete one of the replicates in Table 1. In general, reduced resolvable n-cyclic designs are obtained by deleting one or more replicates from a resolvable n-cyclic design.

We shall first consider the question of which replicates to delete. Let D be the resolvable n-cyclic design and let $D_{ij\ldots}$ be the design obtained by deleting replicates $i,j,\ldots$ from D. Two designs are isomorphic if one can be obtained from the other by a relabelling or permutation of the treatment labels.

Consider the design in Table 1. We can see that replicate 2 is obtained from replicate 1, and replicate 3 from replicate 2, by adding 1, modulo 6, to the levels of the first factor. It follows that D_1, D_2 and D_3 are isomorphic to each other, i.e. $D_1 \sim D_2 \sim D_3$.

More generally, in any resolvable n-cyclic design, a replicate is obtained from the previous replicate by a similar simple permutation. Suppose we have a resolvable n-cyclic design with $r = 4$ replicates and that a 2-replicate design is required. Now under this simple permutation it is clear that there are two equivalence groups of designs given by $D_{12} \sim D_{23} \sim D_{34} \sim D_{14}$ and $D_{13} \sim D_{24}$. Thus we have only two distinct designs, namely D_{12} and D_{13}. Dropping three replicates from a 6-replicate n-cyclic design, we have 20 possible designs. These fall into four equivalence groups, generated from $D_{123}, D_{124}, D_{125}$ and D_{135}, giving four distinct designs. The extension to the general case is straightforward.

The reduced designs are no longer n-cyclic designs; they are simply derived from them. Further, they do not have a factorial structure. Suppose C is a normalised matrix that spans the contrast space of a main effect or interation based on v degrees of freedom. Then the average efficiency factor of this effect is given by

$$E = \frac{v}{r\ \mathrm{trace}(C\Omega C')}. \tag{3}$$

$C\Omega C'\sigma^2$ is the variance–covariance matrix of the estimated effect under the usual additive model for block and treatment parameters, σ^2 is the experimental error, and Ω is a generalised inverse of the information matrix. Thus the average efficiency factor is the ratio of the average variances of the effect in the design ignoring the blocking factors and in the resolvable block design. One choice of C matrix is given by John and Williams (1995), (p. 162).

Two examples of reduced designs obtained by deleting replicates from a full n-cyclic design are given below.

Example 1. Table 2 gives resolvable designs for two factors at 4 and 3 levels in blocks of 4 based on the 2-cyclic design with initial block (00 11 22 31). For a design with $r = 3$ any one replicate can be dropped. For a design with $r = 2$ there are two

Table 2
Resolvable block designs for factors at 4 and 3 levels

r	Replicate deleted	E_1	E_2	E_{12}
4	—	1.0	0.9375	0.6319
3	1	1.0	0.9275	0.5994
2	1, 2	1.0	0.9231	0.4800
2	1, 3	1.0	0.8571	0.5882

Table 3
Resolvable block designs $m_1 = 6$, $m_2 = 4$, $k = 6$ and $r = 3$

Replicate deleted	E_1	E_2	E_{12}
1, 2, 3	1.0	0.9555	0.7451
1, 2, 4	1.0	0.9513	0.7492
1, 2, 5	1.0	0.9513	0.7492
1, 3, 5	1.0	0.9474	0.7461
Partial design	1.0	0.8889	0.5672
Williams–John	1.0	0.9489	0.7652

possibilities and the choice between them is a trade-off between the second main effect and the interaction.

Example 2. Williams and John (1996) considered in some detail the design of an experiment to study the effect of four different pre-treatments on the germination of six *Acacia Mangium* seedlots. A three replicate design was required, with the replicates corresponding to the top, middle and bottom trays of the germination cabinet. A factorial row–column design was eventually used, with the primary aim being to maximise the average efficiency factors of the two main effects. We shall consider how to set out this experiment in blocks of size 6; row–column designs are outside the scope of this paper.

We can construct a resolvable partial 2-cyclic block design for two factors at $m_1 = 6$ and $m_2 = 4$ levels, respectively with $k = 6$ and $r = 3$. The design generated from the initial block (00 11 22 30 41 52) has $E_1 = 1.0$, $E_2 = 0.8889$ and $E_{12} = 0.5672$. Other initial blocks can be chosen, but they have the same efficiency factors.

The alternative is to construct a full resolvable 2-cyclic design with $r = k = 6$ and then delete 3 replicates. The initial block (00 12 21 30 41 53) gives a 6-replicate design with $E_1 = 1.0$, $E_2 = 0.9623$ and $E_{12} = 0.7928$; the best design in terms of first maximising E_1 and E_2, and then maximising E_{12}. Table 3 gives the average efficiency factors of designs obtained by deleting 3 replicates from this design. Also given are the values for the partial design and the best design obtained from running the Williams and John (1996) algorithm a number of times.

Deleting replicates 1, 2 and 3 from the full 2-cyclic design gives the best design in terms of the main effects, although there is some loss in the interaction average efficiency factor compared with the design generated by the Williams and John algorithm. The partial 2-cyclic design does poorly on both E_2 and E_{12}, as expected.

5. Discussion

A flexible class of resolvable block designs for experiments with n factors can be obtained from the family of n-cyclic designs. Such designs have factorial structure, and in general the number of replicates r is equal to k. Partial n-cyclic designs with

$r < k$ can be constructed, but are not generally recommended as some main effects will necessarily be estimated inefficiently. Instead designs with $r < k$ can be derived from the fully resolvable n-cyclic designs by deleting $k - r$ of the replicates.

The only other general method of constructing such designs is the algorithm given by Williams and John (1996). While it ensures that each level of every factor occurs as equally as possible in each block of the design, this in itself does not ensure that the average efficiency factors of the main effects, and the interactions, are necessarily as high as they could be. The resolvable block designs presented in this paper compare favourably with designs generated by this algorithm. Note, however, that the Williams and John algorithm can also generate row–column designs.

Another blocking feature used frequently in practice is latinisation. This is where the replicates are set out under each other. In this case the design can also be viewed as a block design with s long blocks of size rk, where $s = v/k$. Now it is desirable that no treatment combination occurs more than once in any long block. For a discussion of latinised designs see Williams (1986). Because of the cyclic method of construction, resolvable n-cyclic block designs, and the reduced designs derived from them, are automatically latinised. For instance, if the three replicates of the design in Table 1 were contiguous no treatment combination would occur more than once in any of the 8 long blocks of size 9.

Acknowledgements

This work was supported by a grant from the Marsden Fund.

References

John, J.A., Williams, E.R., 1995. Cyclic and Computer Generated Designs. Chapman & Hall, London.

Williams, E.R., 1986. Row and column designs with contiguous replicates. Austral. J. Statist. 28, 154–163.

Williams, E.R., John, J.A., 1996. Row–column factorial designs for use in agricultural field experiments. Appl. Statist. 45, 39–46.

Appendix D

Program for the generation of α_n-designs

```
#include <stdlib.h>
#include <stdio.h>
#include <iomanip.h>
#include <malloc.h>
#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include <time.h>
#include <conio.h>

#define TOPLEVEL    7
#define FALSE       0
#define TRUE        1
#define EPS         0.000001
#define LARGED      99999999.0
#define MAXINT      32000

#define rnd01       (rand()/(double)(RAND_MAX))
#define ran1n(n)    (rand()%n)
#define diter(N,p)  (log(1.0-p)/log((N-1.0)/N))
#define min(a,b)    (((a) < (b)) ? (a) : (b))
#define max(a,b)    (((a) > (b)) ? (a) : (b))
#define Pr(dE, T)   exp(-(dE)/(T))
#define faccept(seqtype,dE,T)   (dE<0 || (rnd01<Pr(dE,T) && seqtype==0))
#define anneal(t,delta,Umax)    (Umax/(1+t*delta))

time_t tstart, tnow;

/*********************************READ_DESIGN_PARAMETERS*********************************/
/*                                                                                      */
/*    This function reads in the design parameters and checks that they satisfy the     */
/*    contraints for an alpha_N-design, i.e. v_i=k_i*s_i.                               */
/*                                                                                      */
/****************************************************************************************/

void read_design_parameters(int &v, int &k, int &r, int &n, int &s,
                            int **vf,int **kf,int **sf, int &seed)
{
    int i,ERROR, ERROR_vf, ERROR_kf;
```

```
ERROR=1;
while(ERROR)
{
    cout << endl << endl
         << "Please enter the design parameters:"
         << endl << endl;
    cout << "Number of treatments,  v = ";
    cin  >> v;
    cout << "Number of units/block, k = ";
    cin  >> k;
    if(!(v % k))ERROR=0;
    else cout << "The number of treatments must be an integer
                  multiple of the block size"
              << endl << endl;
}
cout << "Number of replicates,  r = ";
cin  >> r;
cout << "Number of factors,     n = ";
cin  >> n;
s = v/k;

/**** Read factor levels and plot factors from the command line ****/

/* Allocate memory */
*vf = new int[n];   // factor levels
*kf = new int[n];   // plot factors
*sf = new int[n];   // block factors
/* End memory allocation*/

if(n==1)
{
    (*vf)[0]=v;
    (*kf)[0]=k;
    (*sf)[0]=s;
}
else
{
    cout << "\nPlease enter the number of levels (vi) of factor i, followed by\n";
    cout << "the corresponding plot factor (ki):\n\n" << endl;
    ERROR=1;
    while(ERROR)
    {
        ERROR_vf=ERROR_kf=1;
        for(i=0; i<n; i++)
        {
            cout << "v" << i+1 << ", k" << i+1 << ": ";
            cin  >> (*vf)[i] >> (*kf)[i];
            ERROR_vf*=(*vf)[i];
            ERROR_kf*=(*kf)[i];
            (*sf)[i]=(*vf)[i]/(*kf)[i];
        }
        if(ERROR_vf != v)
        {
            cout << "\nProduct of v[i]'s does not equal " << v << ".  ";
            cout << "Re-enter these parameters.\n" << endl;
            continue;
        }
        if(ERROR_kf != k)
        {
            cout << "\nProduct of kf[i]'s does not equal " << k << ".  ";
            cout << "Re-enter these parameters.\n" << endl;
            continue;
```

```
            }
            ERROR=0;
        }
    }
    cout << endl << "Seed = ";
    cin  >> seed;
}

/************************************** READ_WEIGHTS ********************************/
/*                                                                                    */
/**************************************************************************************/

void read_weights(int n, int n_gint, int **x, int *order_x, double *weight)
{
    int f,m,nway;

    cout << endl;
    nway=1;
    while(nway<4)
    {
        for(m=0; m<n_gint; m++)
        {
            if(order_x[m]==nway)
            {
                if(nway==1)
                {
                    cout << "Weight for Factor ";
                    for(f=0; f<n; f++)if(x[m][f])cout << f+1;
                    cout << " main effect = ";
                }
                else
                {
                    cout << "Weight for ";
                    for(f=0; f<n; f++)if(x[m][f])cout << "F" << f+1;
                    cout << " interaction effect = ";
                }
                cin >> weight[m];
            }
        }
        nway++;
    }
    cout << endl << endl << "Weights for all lower order interactions are set to ZERO.";
    cout << endl << endl;
}

/********************************* SET_UP_INITIAL_ALPHA_ARRAY ************************/
/*                                                                                    */
/**************************************************************************************/

void set_up_initial_alpha_array(int k,int r,int s, int ***alpha)
{
    int i,j;


    /***** Allocate memory to alpha array *****/

    *alpha = new int *[k];
    for(i=0; i<k; i++)(*alpha)[i] = new int [r];

    /****** Generate initial alpha-array ******/

    for(i=0;i<k;i++)
        for(j=0;j<r;j++)(*alpha)[i][j]=rand()%s;
```

```
}

/********************** Write design parameters and e.f.'s to file *********************/

void write_design_parameters(int n, int v, int k, int s, int r, int seed, int *vf,
                             int *kf, int *sf, int **alpha, int n_gint, int **x,
                             double *E, double eff, int *order_x, double *weight)
{
    int f,i,j,m,sum_xf,nway;

    ofstream design_of("design.dat", ios::out);
    if(!design_of)
    {
        cerr << "File could not be opened" << endl;
        exit(1);    // protoype in stdlib.h
    }
    design_of << "Design parameters" << endl;
    design_of << "-----------------" << endl << endl;
    design_of << "v = " << v << ", k = " << k << ", s = " << s << ", r = " << r  << endl;
    for(f=0; f<n; f++)design_of <<   "v" << f+1 << "= " << vf[f]
                                << ", k" << f+1 << "= " << kf[f]
                                << endl;
    design_of << endl << endl;
    design_of << "Weights" << endl;
    design_of << "-------" << endl;
    nway=1;
    while(nway<4)
    {
        for(m=0; m<n_gint; m++)
        {
            if(order_x[m]==nway)
            {
                if(nway==1)
                {
                    design_of << "F";
                    for(f=0; f<n; f++)if(x[m][f])design_of << f+1;
                    design_of << " main effect = ";
                }
                else
                {
                    for(f=0; f<n; f++)if(x[m][f])design_of << "F" << f+1;
                    design_of << " interaction = ";
                }
                design_of << setw(4) << setprecision(2)
                          << setiosflags(ios::fixed | ios::right)
                          << weight[m]
                          << endl;
            }
        }
        nway++;
    }
    design_of << endl << "** Weights for all lower order interactions are set to ZERO.";
    design_of << endl << endl;
    design_of << "Seed = ";
    design_of << setw(3) << setiosflags(ios::left) << seed << endl << endl;
    design_of << "Alpha array" << endl;
    design_of << "-----------" << endl;
    for(i=0; i<k; i++)
    {
        for(j=0; j<r; j++)design_of << setw(3) << setiosflags(ios::right)
                                    << alpha[i][j];
        design_of << endl;
    }
```

```
    design_of << endl;

    design_of << "Average efficiency factors for:" << endl;
    design_of << "-------------------------------" << endl << endl;
    nway=1;
    while(nway<=n)
    {
        if(nway==1)design_of << "Main effects" << endl << endl;
        else       design_of << nway << "-way interactions" << endl << endl;
        for(m=0; m<n_gint; m++)
        {
            sum_xf=0;
            for(f=0; f<n; f++)sum_xf+=x[m][f];
            if(sum_xf==nway)
            {
                design_of << "   E";
                for(f=0; f<n; f++)if(x[m][f])design_of << f+1;
                design_of << " = " << setw(7) << setprecision(5)
                          << setiosflags(ios::fixed | ios::right)
                          << E[m] << endl;
            }
        }
        design_of << endl;
        nway++;
    }

    design_of << endl;
    design_of << "Overall average efficiency factor, E = "
              << setw(7) << setprecision(5) << setiosflags(ios::fixed | ios::right)
              << eff << endl;
    design_of.close();
}

/********************************** WRITE_FACTORIAL_DESIGN ***************************/
/*                                                                                   */
/* This function writes the factorial design to <fact_design.out>.                   */
/*                                                                                   */
/*************************************************************************************/

void write_factorial_design(int n, int r, int k, int s, int *vf, int *sf, int **Z,
                            int **Subgrp, int **bestalpha)
{
    int f,i,j,u;

    ofstream of("fact_design.out", ios::out);
    if(!of)
    {
        cerr << "File could not be opened" << endl;
        exit(1);    // protoype in stdlib.h
    }
    of << "Factorial Design" << endl;
    of << "----------------" << endl << endl;
    of << "Number of factors, n = " << n  << endl;
    for(f=0; f<n; f++)of << "Number of levels of factor " << f+1 << " = " << vf[f]
                         << endl;
    of << "Number of plots per block,      k = " << k << endl;
    of << "Number of blocks per replicate, s = " << s << endl;
    of << "Number of replicates,           r = " << r << endl << endl << endl;

    for(j=0; j<r; j++)
    {
        of << "Replicate " << j+1 << ":" << endl;
        for(i=0; i<k; i++)
```

```
        {
            for(u=0; u<s; u++)
            {
                for(f=0; f<n; f++)
                    of << setw(2) << setiosflags(ios::right)
                       << (Z[f][bestalpha[i][j]]+Z[f][u])%sf[f] + Subgrp[f][i];
                of << "   ";
            }
            of << endl;
        }
    }
    of.close();
}

/********************************** Zgroup *********************************/
/*                                                                         */
/* This function constructs the external direct sum of the subgroups Z_s1, */
/* Z_s2,...,Z_sN.  The first digit of the ith digit (i=1,2,...,s) of the   */
/* jth N-tuple is stored in Z[i-1][j].  The array, Z, will be used in the  */
/* construction of the roots of unity, and again in the construction of the */
/* concurrence vetor.                                                      */
/*                                                                         */
/***************************************************************************/

void Zgroup(int n, int s, int *sf, int ***Z)
{
    int f,i,j,same,check;

    *Z = new int *[n];
    for(f=0; f<n; f++)(*Z)[f] = new int [s];

    same = s;
    for(f=0; f<n; f++)
    {
        check=0;
        same=same/sf[f];
        while(check<s)
        {
            for(i=0; i<sf[f]; i++)
                for(j=0; j<same; j++)(*Z)[f][i*same+j+check]=i;
            check+=sf[f]*same;
        }
    }
}

/********************************** Subgroup *********************************/
/*                                                                           */
/* This function constructs the external direct sum of the subgroups <s1>,   */
/* <s2>,...,<sN>.  The first digit of the ith digit (i=1,2,...,s) of the     */
/* jth N-tuple is stored in Subgrp[i-1][j].  The array, Subgrp, will be used in the */
/* construction of the roots of unity, and again in the construction of the  */
/* concurrence vetor.                                                        */
/*                                                                           */
/*****************************************************************************/

void Subgroup(int n, int k, int *kf, int *sf, int ***Subgrp)
{
    int f,i,j,repeat,check;

    *Subgrp = new int *[n];
    for(f=0; f<n; f++)(*Subgrp)[f] = new int [k];

    repeat = k;
```

```
    for(f=0; f<n; f++)
    {
        check=0;
        repeat = repeat/kf[f];
        while(check<k)
        {
            for(i=0; i<kf[f]; i++)
                for(j=0; j<repeat; j++)(*Subgrp)[f][i*repeat+j+check]=i*sf[f];
            check+=kf[f]*repeat;
        }
    }
}

/********************** Compute and store roots of unity *********************/

void roots_of_unity(int n, int s, int *sf, int **Z, long double ***dcos,
                    long double ***dsin)
{
    int f,hf,uf,i,j;
    long double uh_s, phi, pi=3.1415926535897932385;

    /************* Allocate memory *************/

    *dcos = new long double *[s];
    *dsin = new long double *[s];
    for(i=0; i<s; i++)(*dcos)[i] = new long double [s];
    for(i=0; i<s; i++)(*dsin)[i] = new long double [s];
    for(i=0; i<s; i++)
        for(j=0; j<s; j++)(*dcos)[i][j]=(*dsin)[i][j]=0.0;

    /********* Compute and store roots *********/

    for(uf=0; uf<s; uf++)
        for(hf=0; hf<s; hf++)
        {
            uh_s=0.0;
            for(f=0; f<n; f++)
                uh_s+=(long double)Z[f][uf]*(long double)Z[f][hf]/(long double)sf[f];
            phi=2.0*pi*uh_s;
            (*dcos)[uf][hf]=cos(phi);
            (*dsin)[uf][hf]=sin(phi);
        }
}

/************************************ KRONECKER ************************************/
/*                                                                                 */
/***********************************************************************************/

void kronecker(double **C, double **newC, double **L, int dim_newC, int dim_kf)
{
    int i,j,l,m;


    /* Copy C to newC as C will be overwritten in computing Kronecker product */

    for(l=0; l<dim_newC; l++)
        for(m=0; m<dim_newC; m++)newC[l][m] = C[l][m];

    /* Compute Kronecker product */

    for(l=0; l<dim_newC; l++)
        for(m=0; m<dim_newC; m++)
            for(i=0; i<dim_kf; i++)
```

```
                for(j=0; j<dim_kf; j++)
                    C[l*dim_kf+i][m*dim_kf+j]=newC[l][m]*L[i][j];
}

/***************************** CONSTRUCT_COEFFICIENT_MATRICES ************************/
/*                                                                                   */
/*************************************************************************************/

void construct_coefficient_matrices(int n, int k, int s, int *kf, int *sf, int **u,
                                    int n_gint, int **x, double *****C)
{
    int f,i,j,l,m,kfn,dim_newC;
    double **newC, **L;

    /*************** Allocate memory and initialize arrays ***************/

    *C = new double ***[s];
    for(l=0; l<s; l++)
    {
        (*C)[l] = new double **[n_gint];
        for(m=0; m<n_gint; m++)
        {
            (*C)[l][m] = new double *[k];
            for(i=0; i<k; i++)(*C)[l][m][i] = new double [k];
        }
    }

    for(l=0; l<s; l++)
        for(m=0; m<n_gint; m++)
            for(i=0; i<k; i++)
                for(j=0; j<k; j++)(*C)[l][m][i][j] = 0;

    kfn=k/kf[n-1];
    newC = new double *[kfn];
    for(i=0; i<kfn; i++)newC[i] = new double [kfn];

    L = new double *[k];
    for(i=0; i<k; i++)L[i] = new double [k];

    /************* End allocation of memory and initialization ************/

    for(l=1; l<s; l++)
        for(m=0; m<n_gint; m++)
        {
            dim_newC=1;
            for(f=0; f<n; f++)
            {
                if(u[f][l]==0)
                {
                    if(x[m][f]==0)
                    {
                        for(i=0; i<kf[f]; i++)
                        {
                            /* Factor f absent */
                            for(j=i+1; j<kf[f]; j++)L[i][j]=L[j][i]=1.0/(double)kf[f];
                            L[i][i]=1.0/(double)kf[f];
                        }
                    }
                    else
                    {
                        for(i=0; i<kf[f]; i++)
                        {
                            /* Factor f present */
```

```
                        for(j=i+1; j<kf[f]; j++)L[i][j]=L[j][i]=-1.0/(double)kf[f];
                        L[i][i]=1.0-1.0/(double)kf[f];
                    }
                }
            }
            else
            {
                if(x[m][f]==0)
                {
                    for(i=0; i<kf[f]; i++)
                    {
                        /* Factor f absent */
                        for(j=i+1; j<kf[f]; j++)L[i][j]=L[j][i]=0;
                        L[i][i]=0;
                    }
                }
                else
                {
                    for(i=0; i<kf[f]; i++)
                    {
                        /* Factor f present */
                        for(j=i+1; j<kf[f]; j++)L[i][j]=L[j][i]=0;
                        L[i][i]=1;
                    }
                }
            }
            if(!f)
            {
                for(i=0; i<kf[f]; i++)
                    for(j=0; j<kf[f]; j++)(*C)[l][m][i][j]=L[i][j];
            }
            else
            {
                dim_newC*=kf[f-1];
                kronecker((*C)[l][m], newC, L, dim_newC, kf[f]);
            }
        }
    }

    delete [] L;
    delete [] newC;
}

/****************************** CONSTRUCT_Vx_Kx_Sx ********************************/
/*                                                                                */
/* The average efficiency factors for each main and interaction effect will be    */
/* stored in E[m].  The numerator of each E[m] is given by the product of each    */
/* (vf[f]-1) (f=1...n) for x[m][f]=1.  Similarly, in the denominator of E[m]      */
/* there is the product of the (kf[f]-1) (f=1...n) for x[m][f]=1.  These products */
/* are computed and stored in vx[m] and kx[m], respectively.                      */
/*                                                                                */
/**********************************************************************************/

void construct_vx_kx_sx(int n, int *vf, int *kf, int *sf, int n_gint, int **x,
                        int **vx1, int **kx1, int **sx, int **kx)
{
    int f,m;

    /*************** Allocate memory and initialize arrays ***************/

    *vx1 = new int [n_gint];
    *kx1 = new int [n_gint];
    *sx = new int [n_gint];
```

```
    *kx = new int [n_gint];
    for(m=0; m<n_gint; m++)(*vx1)[m]=(*kx1)[m]=(*sx)[m]=(*kx)[m]=1;

    /************ End allocation of memory and initialization ************/

    for(m=0; m<n_gint; m++)
        for(f=0; f<n; f++)
            if(x[m][f])
            {
                (*vx1)[m] *= (vf[f]-1);
                (*kx1)[m] *= (kf[f]-1);
                (*sx)[m] *= sf[f];
            }
            else (*kx)[m] *= kf[f];
}
/******************************** DOUBLETRANS ********************************/
/*                                                                           */
/* This function copies A to newA.  When A is used by NEWSWEEP it is        */
/* destroyed, and would therefore be unavailable for later use.              */
/*                                                                           */
/*****************************************************************************/

void doubletrans(int k, double **A, double **newA)
/* This routine sets newA to A */
{
    int i,j;

    for(i=0; i<k; i++)
        for(j=0; j<k; j++)newA[i][j]=A[i][j];
}

/******************************** NEWSWEEP ***********************************/
/*                                                                           */
/* In this function v = block size and A = Re(Astar) = t1.  The function     */
/* computes the inverse of A and destroys the upper triangle of this matrix */
/* by replacing the elements in the upper triangle with the elements of the */
/* upper triangle from A^(-1).                                               */
/*                                                                           */
/*****************************************************************************/

void newsweep(int v, double **A, double *x, int k, double *det)
{
    int i,j;
    double  D,B,C;

    D=A[k][k];
    if(D<EPS)*det=0.0;
    else
    {
        *det = *det*D;
        for(i=0; i<v; i++)
            if(i!=k)
            {
                if(i<k)B=A[i][k]/D;
                else   B=x[i]*x[k]*A[k][i]/D;
                for(j=i; j<v; j++)
                    if(j!=k)
                    {
                        if(k<j)C=A[k][j];
                        else   C=x[j]*x[k]*A[j][k];
                        A[i][j]=A[i][j]-B*C;
                    }
            }
```

```
            for(i=0; i<k; i++)A[i][k]=-A[i][k]/D;
            for(j=k; j<v; j++)A[k][j]= A[k][j]/D;
            A[k][k]=1.0/D;
            x[k]=-x[k];
    }
}

/********************************** VECTOR **********************************/

void vector (double **A, double *x, double *y, int k)
{
    int i,j,l;

    for(l=0; l<k; l++)
    {
        y[l]=0.0;
        for (i=0; i<=l; i++)  y[l]=y[l]+x[i]*A[i][l];
        for (j=l+1; j<k; j++) y[l]=y[l]+x[j]*A[l][j];
    }
}

/****************************** CONCURRENCE_MATRIX ******************************/
/*                                                                              */
/********************************************************************************/

void concurrence_matrix(int n, int v, int k, int s, int r, int *sf, int **Z, int **alpha,
                        int **NN)
{
    int f,i,j,l,x1,x2,ij1,ij2,ss;

    l=0;
    while(l<k)
    {
        NN[l][l*s]=r;
        for(i=l+1; i<k; i++)
        {
            for(j=0; j<r; j++)
            {
                x1=x2=ij1=ij2=0;
                ss=s;
                for(f=0; f<n; f++)
                {
                    ij1=Z[f][alpha[i][j]]-Z[f][alpha[l][j]];
                    if(ij1<0)ij1=sf[f]+ij1;
                    ij2=Z[f][alpha[l][j]]-Z[f][alpha[i][j]];
                    if(ij2<0)ij2=sf[f]+ij2;
                    if(f<n-1)
                    {
                        ss/=sf[f];
                        x1 += (ij1*ss);
                        x2 += (ij2*ss);
                    }
                    else
                    {
                        x1 += ij1;
                        x2 += ij2;
                    }
                }
                NN[l][i*s+x1]++;
                NN[i][l*s+x2]++;
            }
        }
        l++;
```

```
    }
}

/******************************** UPDATE_CONCURRENCE_MATRIX ****************************/
/*                                                                                      */
/****************************************************************************************/

void update_concurrence_matrix(int n, int k, int s, int *sf, int **Z, int **alpha,
                               int **NN, int row, int col, int oldalpha)
{
    int f,h,i,ij1,ij2,x1,x2,ss,sign1,sign2,a[2];

    ij1=ij2=0;
    a[0]=oldalpha;
    a[1]=alpha[row][col];
    for(i=0; i<k; i++)
    {
        if(i<row)sign1= 1;
        if(i>row)sign1=-1;
        sign2=-1;
        for(h=0; h<2 && i!=row; h++)
        {
            x1=x2=0;
            ss=s;
            for(f=0; f<n; f++)
            {
                ij1=(Z[f][a[h]]-Z[f][alpha[i][col]])*sign1;
                if(ij1<0)ij1=sf[f]+ij1;
                ij2=(sf[f]-ij1)%sf[f];
                if(f<n-1)
                {
                    ss/=sf[f];
                    x1+=(ij1*ss);
                    x2+=(ij2*ss);
                }
                else
                {
                    x1+=ij1;
                    x2+=ij2;
                }
            }
            if(i<row)
            {
                NN[i][row*s+x1]=NN[i][row*s+x1]+sign2;
                NN[row][i*s+x2]=NN[row][i*s+x2]+sign2;
            }
            else
            {
                NN[row][i*s+x1]=NN[row][i*s+x1]+sign2;
                NN[i][row*s+x2]=NN[i][row*s+x2]+sign2;
            }
            sign2=-sign2;
        }
    }
}
/***************************** END update_concurrence_matrix ************************/

/********************** GENERALIZED INTERACTION INDICES ***********************/
/*                                                                            */
/******************************************************************************/

void interaction_indices(int n, int n_gint, int ***x, int **order_x)
{
```

```
    int i,f,l,m,maineff;

    *x = new int *[n];
    for(i=0; i<n_gint; i++)(*x)[i] = new int [n];
    for(i=0; i<n_gint; i++)
        for(f=0; f<n; f++)(*x)[i][f]=0;
    (*order_x) = new int [n_gint];
    for(i=0; i<n_gint; i++)(*order_x)[i]=0;

    maineff=l=0;
    while(maineff<n)
    {
        (*x)[l++][maineff++]=1;
        m=l-1;
        for(i=0; i<m; i++)
        {
            for(f=0; f<n; f++)(*x)[l][f]=(*x)[i][f]+(*x)[m][f];
            l++;
        }
    }
    for(i=0; i<n_gint; i++)
        for(f=0; f<n; f++)(*order_x)[i]+=(*x)[i][f];
}

/********************** Compute X1 (the real part of Omega.star) **********************/
/*****************************************************************************************/

void neweff(int n, int v, int k, int s, int r, long double c3, long double c4, int *vf,
             int *kf, int *sf, int **NN, long double **dcos, long double **dsin,
             int **alpha, double ****C, int *vx1, int *kx1, int n_gint, int **x,
             double *eff, double *E)
{
    int i,j,l,m,u;
    double sing, sum;
    double *z, **t1, **t2, **X;


    z = new double [k];
    X  = new double *[k];
    t1 = new double *[k];
    t2 = new double *[k];
    for(i=0; i<k; i++)
    {
        t1[i] = new double [k];
        t2[i] = new double [k];
        X[i]  = new double [k];
    }

    for(i=0; i<n_gint; i++)E[i]=0.0;

    sum = 0.0;
    sing=1.0;
    u=1;
    while(sing>EPS && u<s)
    {
        /*****************************************************/
        /* Initialize only the upper triangles of t1 and t2 */
        /* to zero since t1=Re{A*(u)} is symmetric and      */
        /* t2=Im{A*(u)} is skew-symmetric.  Therefore, the  */
        /* lower triangles can be deduced from their        */
        /* respective upper triangles.                      */
        /*****************************************************/
```

```
for(i=0;i<k;i++)
    for(j=i+1;j<k;j++)
    {
        t1[i][j]=0;
        t2[i][j]=0;
    }

/* BEGIN construction of t1 and t2. */

for(l=0;l<s;l++)
{
    for(i=0;i<k;i++)
        for(j=i+1;j<k;j++)
        {
            t1[i][j]=t1[i][j]+dcos[u][l]*NN[i][j*s+l];
            t2[i][j]=t2[i][j]+dsin[u][l]*NN[i][j*s+l];
        }
}
for(i=0;i<k;i++)
    for(j=i;j<k;j++)if(i==j)
                    {
                        t1[i][j]=c4;
                        t2[i][j]=0;
                    }
                    else
                    {
                        t1[i][j]=-c3*t1[i][j];
                        t2[i][j]=-c3*t2[i][j];
                    }

for(i=1;i<k;i++)
    for(j=0;j<i;j++)
    {
        t1[i][j]= t1[j][i];
        t2[i][j]=-t2[j][i];
    }

/* END construction of t1 and t2. */

doubletrans(k, t1, X); /* Copy t1 to X. t1 is destroyed by NEWSWEEP. */
for(i=0; i<k; i++)z[i]=1.0;
sing=1.0;
i=0;
while(sing>EPS && i<k)
{
    newsweep(k, t1, z, i, &sing);
    i++;
}
if (sing>EPS)
{
    for(i=0; i<k; i++)
    {
        vector(t1,t2[i],z,k);
        for(j=i; j<k; j++)
            for(l=0; l<k; l++)X[i][j]=X[i][j]+z[l]*t2[l][j];
    }

    for(i=0;i<k;i++)z[i]=1.0;
    sing=1.0;
    i=0;
    while(sing>EPS && i<k)
    {
        newsweep(k,X,z,i,&sing);
```

```
                i++;
            }

            /************************ Compute efficiency factors ************************/

            for(i=0; i<k; i++)sum+=X[i][i];
            for(m=0; m<n_gint; m++)
                for(l=0; l<k; l++)
                {
                    for(i=0; i<=l; i++) E[m]+=(C[u][m][i][l]*X[i][l]);
                    for(j=l+1; j<k; j++)E[m]+=(C[u][m][l][j]*X[l][j]);
                }
            /*****************************************************************************/
        }
        u++;
    }
    if(sing<EPS)
    {
        *eff=0.0;
        for(m=0; m<n_gint; m++)E[m]=0.0;
    }
    else
    {
        *eff=(double)(v-1.0)/(double)((k-1.0)+r*sum);
        for(m=0; m<n_gint; m++)E[m] = (double)vx1[m]/((double)kx1[m]+(double)r*E[m]);
    }

    delete [] z;
    for(i=0; i<k; i++)
    {
        delete t1[i];
        delete t2[i];
        delete X[i];
    }
    delete [] t1;
    delete [] t2;
    delete [] X;
}

/********************************** OBJECTIVE FUCTION **********************************/
/*                                                                                     */
/* This objective function is based on a weighted average of the average e.f.'s, up to */
/* and including 3-way interactions.                                                   */
/*                                                                                     */
/***************************************************************************************/

double objective(int n_gint, int *order_x, double *weight, double *E)
{
    int i;
    double wt_efx;

    wt_efx=0.0;
    for(i=0; i<n_gint; i++)if(order_x[i]<4)wt_efx+=(weight[i]*E[i]);

    return wt_efx;
}

/****************************** CONSTRUCT_INDICATOR_MATRICES ***************************/
/*                                                                                     */
/***************************************************************************************/

void construct_indicator_matrices(int n, int s, int k, int *kf, int n_gint, int **x,
                                  int **Z, int ***In_Tx, int ****In_trCxNN)
```

```
{
    int f,i,i1,j,j1,m,u,ng,dim_oldIn;
    int ***I, ***J; // Identity matrix and square matrix of 1's
    int **oldIn;    // Indicator matrix: identifies elements to sum for trCxNN

    // Allocate memory to I_{k_i} and J_{k_i} ...

    I = new int **[n];
    J = new int **[n];
    for(f=0; f<n; f++)
    {
        I[f] = new int *[kf[f]];
        J[f] = new int *[kf[f]];
        for(i=0; i<kf[f]; i++)
        {
            I[f][i] = new int [i+1];
            J[f][i] = new int [i+1];
        }
    }
    // Set up I and J...
    for(f=0; f<n; f++)
        for(i=0; i<kf[f]; i++)
        {
            I[f][i][i] = J[f][i][i] = 1;
            for(j=i+1; j<kf[f]; j++)
            {
                I[f][j][i] = 0;
                J[f][j][i] = 1;
            }
        }

    // Allocate memory to kxk indicator matrix, In_trCxNN, temporary storage for
    // indicator matrix, oldIn, and the T_u1u2...uN indicator matrix, In_Tx.  The
    // latter indicator matrix is used to identify which T_u1u2...uN matrices are
    // summed over and depend on the generalized interaction, x=x1x2...xN, of
    // interest.

    ng = n_gint-1;
    *In_Tx = new int *[ng];
    *In_trCxNN = new int **[ng];
    oldIn = new int *[k];
    for(m=0; m<ng; m++)
    {
        (*In_Tx)[m] = new int [s];
        for(u=0; u<s; u++)(*In_Tx)[m][u]=1;  // Initialize In_Tx
        (*In_trCxNN)[m] = new int *[k];
        for(i=0; i<k; i++)
        {
            (*In_trCxNN)[m][i] = new int [i+1];
            oldIn[i] = new int [i+1];
        }
    }

    // Construct indicator matrices...
    for(m=0; m<ng; m++)
    {
        // (Re-)Initialize oldIn...
        for(i1=0; i1<k; i1++)
            for(j1=0; j1<i1+1; j1++)
                (*In_trCxNN)[m][i1][j1]=oldIn[i1][j1]=1;
        dim_oldIn=1;
        for(f=0; f<n; f++)
        {
```

```
            if(x[m][f])
            {
                for(u=0; u<s; u++)if(Z[f][u])(*In_Tx)[m][u]=0;
                for(i1=0; i1<dim_oldIn; i1++)
                    for(j1=0; j1<i1+1; j1++)
                        for(i=0; i<kf[f]; i++)
                            for(j=0; j<i+1; j++)
                            {
                                (*In_trCxNN)[m][i1*kf[f]+i][j1*kf[f]+j]=
                                                        oldIn[i1][j1]*I[f][i][j];
                                if(j1<i1)(*In_trCxNN)[m][i1*kf[f]+j][j1*kf[f]+i]=
                                                        oldIn[i1][j1]*I[f][i][j];
                            }
            }
            else
            {
                for(i1=0; i1<dim_oldIn; i1++)
                    for(j1=0; j1<i1+1; j1++)
                        for(i=0; i<kf[f]; i++)
                            for(j=0; j<i+1; j++)
                            {
                                (*In_trCxNN)[m][i1*kf[f]+i][j1*kf[f]+j]=
                                                        oldIn[i1][j1]*J[f][i][j];
                                if(j1<i1)(*In_trCxNN)[m][i1*kf[f]+j][j1*kf[f]+i]=
                                                        oldIn[i1][j1]*J[f][i][j];
                            }
            }
            dim_oldIn*=kf[f];
            for(i1=0; i1<dim_oldIn; i1++)
                for(j1=0; j1<i1+1; j1++)oldIn[i1][j1]=(*In_trCxNN)[m][i1][j1];
        }
    }

    delete [] I;
    delete [] J;
    delete [] oldIn;
}
/***************************** END construct_indicator_matrices ***********************/

/************************************ SET_SEQTYPE *************************************/
/*                                                                                    */
/**************************************************************************************/

void set_seqtype(long iterseq, int *seqtype, int *row, int *col, int *alpha_new,
                 long *iter, int s)
{
    *seqtype=TRUE;
    *row=0;
    *col=0;
    *alpha_new=s-1;
    *iter=iterseq;
}

/************************************ PERTURBATION ************************************/
/*                                                                                    */
/* This routine moves to a neigbouring alpha array.                                   */
/*                                                                                    */
/**************************************************************************************/

void perturb(int seqtype,int r,int k,int s,int **alpha,int *row,int *col,int *alpha_old,
             int *alpha_new)
{
    if (!seqtype)
```

```
    {
        *row=ran1n(k);                      // Choose a random alpha j -> j1 (0 .. k-1)
        *col=ran1n(r);                      // Choose a random alpha i -> i1 (0 .. r-1)
        do    *alpha_new=ran1n(s);          // Choose random array value -> a(0,...,s-1)
        while (*alpha_new == *alpha_old);   // (different to current one)
    }
    else
    {
        if(*alpha_new<s-1)*alpha_new=*alpha_new+1;
        else if (*row<k-1)
        {
            *row=*row+1;
            *alpha_new=0;
        }
        else if (*col<r-1)
        {
            *row=0;
            *col=*col+1;
            *alpha_new=0;
        }
        else
        {
            *row=0;
            *col=0;
            *alpha_new=0;
        }
    }
    *alpha_old=alpha[*row][*col];
    alpha[*row][*col]=*alpha_new;
}
/*********************************** END perturbation ******************************/

/************************************** FINDTEMP ***********************************/
/*                                                                                  */
/************************************************************************************/

void findtemp(int n, int v, int r, int s, int k, int *vf, int *sf, int *kf, int *vx1,
              int *kx1, int *sx, int *kx, int n_gint, int **alpha, int **NN, int **x,
              int **Z,int **In_Tx, int ***In_trCxNN, double c3, double c4,
              double *maxdrop, double *mindrop, double *best_efx, long double **dcos,
              long double **dsin, double ****C, double *eff, double *E, int *order_x,
              double *weight)
{
    int i,row,col,alpha_old,alpha_new;
    double last_efx,wt_efx,diff;

    *maxdrop=0.0;
    *mindrop=LARGED;
    last_efx=*best_efx;
    for(i=0; i<100; i++)
    {
        perturb(FALSE,r,k,s,alpha,&row,&col,&alpha_old,&alpha_new);
        update_concurrence_matrix(n,k,s,sf,Z,alpha,NN,row,col,alpha_old);
        neweff(n,v,k,s,r,c3,c4,vf,kf,sf,NN,dcos,dsin,alpha,C,vx1,kx1,n_gint,x,eff,E);
        wt_efx=objective(n_gint, order_x, weight, E);
        diff=fabs(wt_efx-last_efx);
        if(*maxdrop<diff)*maxdrop=diff;
        if(*maxdrop>diff && diff>0.01)*mindrop=diff;
        last_efx=wt_efx;
    }
    *best_efx=last_efx;
}
/********************************** END findtemp ***********************************/
```

```
/************************************ CALC_LF ****************************************/
/*************************************************************************************/

double calc_LF(int n,double T,double U,double lowprob)
{
    double LF;

    LF=pow(-log(lowprob)*T/U, 1.0/n);

    return LF;
}

/************************************ INTTRANS  **************************************/
/* This routine sets newA to A                                                       */
/*************************************************************************************/

void inttrans(int k,int r,int **newA,int **A)
{
    int i,j;

    for(i=0; i<k; i++)
        for(j=0; j<r; j++)newA[i][j]=A[i][j];
}

/******************************** WRITE_SHORT_INFO ************************************/
/*                                                                                   */
/*************************************************************************************/

void write_short_info(double best_efx, double eff)
{
//  int i;
//  int *hist;

//  hist = new int [r*r+1];
//  for(i=0;i<s*r*(r-1)/2;i++)hist[nn[i]]++;
    tnow=time(NULL);

    cout<< setw(11) << setprecision(7) << setiosflags(ios::fixed | ios::showpoint)
        << best_efx
        << setw(11) << setprecision(7) << setiosflags(ios::fixed | ios::showpoint)
        << eff
        << setw(8) << setiosflags(ios::right) << tnow-tstart;
//  for(i=0; i<=r*r; i++)if(hist[i]>0)cout << setw(3) << setiosflags(ios::right)
//                                         << i
//                                         << setw(4) << setiosflags(ios::right)
//                                         << hist[i];
    cout << endl;

//  delete hist;
}

/************************************ UPLEVEL ****************************************/
/*                                                                                   */
/*************************************************************************************/

void uplevel(long iter, double LF, double DF, double avgtr_x, long *bestlevel,
                int *levelt, long *n, int *level, double *delta, double *levelT,
                double *T, int *t, double *Umax, double *levelU)
/* Go up a level and store parameters */
{
    bestlevel[*level]=avgtr_x;
    levelt[*level] = *t;
```

```
    levelT[*level] = *T;
    *t=1;
    *T = *T/LF;
    *delta = *delta*DF;
    (*level)++;
//  printf("UPLEVEL %d  %lf\n",*level,*T);
    *Umax=levelU[*level];
    if(*level > TOPLEVEL)
    {
        printf("\n\n    ***LEVEL %d TOO HIGH FOR PROGRAM DIMENSION***\n",*level);
        printf("Temperature %lf\n",*T);
        exit(1);
    }
    *n=iter;
}

/************************************ DOWNLEVEL ***************************************/
/*                                                                                    */
/**************************************************************************************/

void downlevel(double DF,int *levelt,long *bestlevel,int *level,int *t,
                    double *levelT,double *T,double *delta,double *avgtr_x,double *Umax,
                    double *levelU)
/* Go down a level and restore parameters */
{
    int i;

    (*level)--;
//  printf("DOWNLEVEL %d  %lf\n",*level,*T);
    *delta = *delta/DF;
    *T = levelT[*level];
    *t=levelt[*level]+1;
    *avgtr_x=bestlevel[*level];
    *Umax=levelU[*level];
    for(i=*level+1; i<TOPLEVEL; i++)bestlevel[i]=99999999L;
}

/******************************** NEST_ANNEAL_ALPHA ***********************************/
/*                                                                                    */
/**************************************************************************************/

void nest_anneal_alpha(int n, int v, int k, int s, int r, int seed, int n_gint, int *vf,
                       int *kf, int *sf, int *vx1, int *kx1, int *sx, int *kx, int **Z,
                       int **x,int **alpha, int **NN, int **In_Tx, int ***In_trCxNN,
                       double ****C, long double **dcos, long double **dsin,
                       long iterrnd, long iterseq, double T, double Umax, double Umin,
                       double delta, double DF, double best_efx, long double c3,
                       long double c4, double *E, double *weight, int *order_x,
                       int **Subgrp)
{
    int alpha_old,alpha_new,row,col;
    int t=0;                        // Count used for temperature decrease
    int level=0;                    // Annealing level, higher levels have lower temp
    int accept;
    int i,j;                            // Local count variable
    int noexit=TRUE;                // True until best design or T cooled at level 0
    int seqtype;                    // True when for sequential perturbations
    int onetemp;                    // True when annealed for one temprature at a level
    int *levelt;                    // Stores t at each level
    int **bestalpha;                // Stores best alpha-array
    long iter;                      // Set at iterseq or iterrnd
    long nit;                       // Iteration count variable
    long *bestlevel;                // Array to store best t2A found at each level
```

```
double eff;                     // Average e.f. of unstructured treatments
double besteff;                 // Best average e.f. of unstructured treatments
double dE;                      // Change in objective after perturbation
double LF;                      // Divides into T for supercooling
double lowprob2;                // Lower limit to go up levels
double upper_eff;               // Upper bound for efficiency factor
double avg_efx,last_efx;        // Efficiency factor of design
double *levelT;                 // Stores T2 at each level
double *levelU;                 // Stores Umax at each level
double *bestE;                  // Stores Ex for best alpha-array

bestE = new double [n_gint];
bestalpha = new int *[k];
for(i=0; i<k; i++)bestalpha[i] = new int [r];

levelt = new int [TOPLEVEL];
levelT = new double [TOPLEVEL];
levelU = new double [TOPLEVEL];

bestlevel = new long [TOPLEVEL];
for(i=0; i<TOPLEVEL; i++)bestlevel[i]=99999999L;

upper_eff=0.0;
for(i=0; i<n_gint; i++)upper_eff+=weight[i];
lowprob2=1E-9;

LF=calc_LF(TOPLEVEL,T,Umin,lowprob2);
for(i=0; i<TOPLEVEL; i++)
{
    levelU[i]=Umax;
    Umax=Umax/LF;
}
Umax=levelU[0];

last_efx=best_efx;
write_short_info(best_efx,eff); // check what hist[] does in this prototype
while(noexit)
{
    seqtype=FALSE;
    iter=iterrnd;
    onetemp=FALSE;
    accept=TRUE;
    while(noexit && !seqtype && accept)
    {
        if(Pr(Umin,T/LF)<lowprob2 && !seqtype)
            set_seqtype(iterseq,&seqtype,&row,&col,&alpha_new,&iter,s);
        accept=FALSE;
        t++;
        T=anneal(t,delta,Umax);
        for(nit=0; nit<iter && noexit; nit++)
        {
            perturb(seqtype,r,k,s,alpha,&row,&col,&alpha_old,&alpha_new);
            if(alpha_old!=alpha_new)
            {
                update_concurrence_matrix(n,k,s,sf,Z,alpha,NN,row,col,alpha_old);
                neweff(n,v,k,s,r,c3,c4,vf,kf,sf,NN,dcos,dsin,alpha,C,vx1,kx1,n_gint,x,
                       &eff,E);
                avg_efx = objective(n_gint, order_x, weight, E);
                dE=last_efx-avg_efx;
                if(faccept(seqtype,dE,T))
                {
                    last_efx=avg_efx;
                    accept=TRUE;
```

```
                    if(avg_efx>best_efx+EPS)
                    {
                        best_efx=avg_efx;
                        for(i=0; i<k; i++)
                            for(j=0; j<r; j++)bestalpha[i][j] = alpha[i][j];
                        for(i=0; i<n_gint; i++)bestE[i] = E[i];
                        besteff = eff;
                        write_short_info(best_efx,eff);
                    }
                    if(fabs(dE)>0 && seqtype)nit=0;
                }
                else
                {
                    alpha[row][col]=alpha_old;
                    update_concurrence_matrix(n,k,s,sf,Z,alpha,NN,row,col,alpha_new);
                    neweff(n,v,k,s,r,c3,c4,vf,kf,sf,NN,dcos,dsin,alpha,C,vx1,kx1,
                           n_gint,x,&eff,E);
                }
                if(nit>=min(500,iter-1))onetemp=TRUE;
                if(last_efx < bestlevel[level] && onetemp && !seqtype)
                {
                    uplevel(iter,LF,DF,avg_efx,bestlevel,levelt,&nit,&level,&delta,
                            levelT, &T,&t,&Umax,levelU);
                    onetemp=FALSE;
                    accept=TRUE;
                }
                noexit=!(upper_eff-best_efx < EPS);
            }
            if(kbhit())noexit=FALSE;
        }/* end of nit loop */
    }/* end of while loop */
    if(level==0)noexit=FALSE;
    else
    {
        downlevel(DF,levelt,bestlevel,&level,&t,levelT,&T,&delta,&last_efx,&Umax,
                  levelU);
        T=anneal(t,delta,Umax);
    }
}/* end of while loop */
write_design_parameters(n,v,k,s,r,seed,vf,kf,sf,bestalpha,n_gint,x,bestE,besteff,
                        order_x,weight);
write_factorial_design(n,r,k,s,vf,sf,Z,Subgrp,bestalpha);

delete [] levelT;
delete [] levelt;
delete [] levelU;
delete [] bestlevel;
delete [] E;
delete [] bestE;
}

/************************************ MAIN ***************************************/
/*                                                                               */
/*********************************************************************************/

void main()
{

    //************************* Declare global variables **********************//

    char str[5];

    int i,j;                // Loop index
```

```
int v;              // No. of factorial treatment combinations
int k;              // No. of plots per block
int s;              // No. of blocks per rep
int r;              // No. of reps
int n;              // No. of factors
int seed;           // Seed for random starting alpha_N generating array
int n_gint;         // No. of generalized interactions
int *vf;            // Vector of factor levels,ie. vf[i]=# levels factor i(i=0...n-1)
int *kf;            // Vector of plot factors
int *sf;            // Vector of block factors
int **NN;           // Vector of treatment concurrences
int *order_x;       // Order of interactions: 1=main effect, n=n-way interaction
int *vx1;           // Product of vf[f]-1; dependent on x1x2...xN
int *kx1;           // Product of kf[f]-1; dependent on x1x2...xN
int *sx;            // Product of sf[f]; dependent on x1x2...xN
int *kx;            // Product of kf[f]; dependent on x1x2...xN
int **alpha;        // k x r alpha array
int **Z;            // n x s array: Z_{s_1} otimes Z_{s_2} otimes ... otimes Z_{s_n}
int **Subgrp;       // n x k array: <s_1> otimes <s_2> otimes ... otimes <s_n>
int **x;            // Pointers to 2-D integer arrays
int **In_Tx;        // Indicator matrix for T_u1u2...uN matrices
int ***In_trCxNN;   // Indicator matrix for trace(CxNN')

double eff;
double best_efx;    // Best average tr(CxNN') from findtemp()
double delta=0.3;   // Anneal Temp. parameter, large cools faster
double DF=1.0;      // Multiplies delta as level increases
double Umax;        // Anneal temp. parameter, measure of max obj. change
double Umin;        // Measure of min obj. change > 0
double T;           // Annealing temperature
double *weight;     // Weights of main/int'n effects for objective function
double *E;
double ****C;       // Contrast matrices

long iterseq;       // Iterations for sequential perturbations
long iterrnd;       // Iterations for random perturbations

long double c3,c4;
long double **dcos; // Real roots of unity
long double **dsin; // Complex roots of unity

str[0]='\0';

read_design_parameters(v, k, r, n, s, &vf, &kf, &sf, seed);
srand((unsigned)seed);
set_up_initial_alpha_array(k, r, s, &alpha);
n_gint=(int)pow(2,n)-1;
interaction_indices(n,n_gint,&x,&order_x);
weight = new double [n_gint];
for(i=0; i<n_gint; i++)weight[i]=0.0;
read_weights(n,n_gint,x,order_x,weight);
tstart=time(NULL);

// Set up initial subgroup, indicator & coeff matrices, etc.
// needed for later computations...

Zgroup(n,s,sf,&Z);
Subgroup(n,k,kf,sf,&Subgrp);

construct_coefficient_matrices(n, k, s, kf, sf, Z, n_gint, x, &C);
construct_vx_kx_sx(n, vf, kf, sf, n_gint, x, &vx1, &kx1, &sx, &kx);
roots_of_unity(n, s, sf, Z, &dcos, &dsin);
```

```
    c3=1.0/k;
    c4=(long double)r*(1.0-1.0/k);
    NN = new int *[k];
    for(i=0; i<k; i++)
    {
        NN[i] = new int [v];
        for(j=0; j<v; j++)NN[i][j]=0;
    }
    E = new double [n_gint];
    concurrence_matrix(n, v, k, s, r, sf, Z, alpha, NN);
    construct_indicator_matrices(n, s, k, kf, n_gint, x, Z, &In_Tx, &In_trCxNN);

    iterseq=r*k*s;
    iterrnd=diter(iterseq, 0.70); // Covers 70%
    findtemp(n,v,r,s,k,vf,sf,kf,vx1,kx1,sx,kx,n_gint,alpha,NN,x,Z,In_Tx,In_trCxNN,c3,c4,
             &Umax,&Umin,&best_efx,dcos,dsin,C,&eff,E,order_x,weight);
    if(Umin>Umax)Umin=1.0;
    T=max(10.0*Umax,200);
    Umax=T;
    cout << " Avg. Eff_x        Eff    Time" << endl;
    nest_anneal_alpha(n,v,k,s,r,seed,n_gint,vf,kf,sf,vx1,kx1,sx,kx,Z,x,alpha,NN,In_Tx,
                      In_trCxNN,C,dcos,dsin,iterrnd,iterseq,T,Umax,Umin,delta,DF,
                      best_efx,c3,c4,E,weight,order_x,Subgrp);
    cout << endl << endl << endl << endl << "Program finished!" << endl;
}
```

References

Anton, H. (1987) *Elementary linear algebra.* Wiley, New York.

Cotter, S. C., John, J. A. and Smith, T. M. F. (1973) Multi-factor experiments in non-orthogonal designs. *J. R. Statist. Soc.* B, **35**, 361–367.

David, H. A. (1967) Resolvable cyclic designs. *Sankhyā*, **29**, 191–198.

Davis, A. W. and Hall, W. B. (1969) Cyclic change-over designs. *Biometrika*, **56**, 283–293.

Dean, A. M. and Lewis, S. M. (1983) Upper bounds for average efficiency factors of two-factor interactions. *J. R. Statist. Soc.* B, **45**, 252–257.

Dean, A. M. and Voss, D. T. (1999) *Design and Analysis of Experiments.* Springer, New York.

Eccleston, J. A. and Hedayat, A. (1974) On the theory of connected designs: characterization and optimality. *Ann. Statist.*, **2**, 1238–1255.

Eccleston, J. A. and Whitaker, D. (1999) On the design of optimal change-over experiments through multi objective simulated annealing. *Statistics and Computing*, **9**, 37–42.

Fisher, R. A. (1926) The arrangement of field experiments. *Journal of the Ministry of Agriculture of Great Britain*, **33**, 503–513.

Fisher, R. A. (1935) *The Design of Experiments.* Oliver and Boyd, Edinburgh.

Fraleigh, J. B. (1982) *A First Course in Abstract Algebra.* Addison-Wesley, Massachusetts.

Graybill, F. A. (1987) *Matrices with Applications in Statistics.* Wadsworth, Belmont.

Harshbarger, B. (1947) Rectangular lattices. *Virginia Agricultural Experimental Station Memoir 1.*

Jarrett, R. G. and Hall, W. B. (1978) Generalized cyclic incomplete block designs. *Biometrika*, **65**, 397–401.

John, J. A. (1973) Generalised cyclic designs in factorial experiments. *Biometrika*, **60**, 55–63.

John, J. A. and Ruggiero, K. (1999) Resolvable block designs for factorial experiments. *J. Statist. Planng Inf.*, **77**, 293–299.

John, J. A. and Smith, T. M. F. (1972) Two-factor experiments in non-orthogonal designs. *J. R. Statist. Soc.* B, **34**, 401–409.

John, J. A. and Whitaker, D. (1993) Construction of resolvable row-column designs using simulated annealing. *Austral. J. Statist.*, **35**, 237–245.

John, J. A. and Whitaker, D. (2000) Recursive formulae for the average efficiency factor in block and row-column designs. *Austral. J. Statist.*, **35**, 237–245.

John, J. A. and Williams, E. R. (1995) *Cyclic and computer generated designs.* Chapman & Hall, London.

John, J. A. and Williams, E. R. (1999) Construction of resolvable designs with nested treatment structure. *Biometrical Journal*, **41**, 341–349.

Kuehl, R. O. (2000) *Design of experiments: Statistical principles of research design and analysis.* Duxbury, Pacific Grove, CA, 2nd edition.

Lewis, S. M. and Dean, A. M. (1980) Factorial experiments in resolvable generalized cyclic designs. *Bulletin in Applied Statistics*, **7**, 159–167.

Lewis, S. M. and Dean, A. M. (1984) Upper bounds for factorial efficiency factors. *J. R. Statist. Soc.* B, **46**, 273–278.

Mukerjee, R. (1979) Inter-effect-orthogonality in factorial experiments. *Calcutta Statist. Assoc. Bull.*, **28**, 83–108.

Patterson, H. D. and Williams, E. R. (1976) A new class of resolvable incomplete block designs. *Biometrika*, **63**, 83–92.

Searle, S. R. (1971) *Linear Models.* J. Wiley & Sons, New York.

Searle, S. R. (1982) *Matrix Algebra Useful for Statistics.* J. Wiley & Sons, New York.

Shah, B. V. (1959) A generalization of partially balanced incomplete block designs. *Ann. Math. Statist.*, **30**, 1041–1050.

Shah, K. R. (1960) Optimality criteria for incomplete block designs. *Ann. Math. Statist.*, **31**, 791–794.

Whitaker, D., Williams, E. R. and John, J. A. (1997) *CycDesigN: Experimental Design Package.* CSIRO Canberra.

Williams, E. R. (1975) *A new class of resolvable block designs.* Ph.D. thesis, University of Edinburgh.

Williams, E. R. (1986) Row and column designs with contiguous replicates. *Austral. J. Statist.*, **28**, 154–163.

Williams, E. R. and John, J. A. (1996) Row-column factorial designs for use in agricultural field trials. *App. Statist.*, **45**, 39–46.

Williams, E. R. and John, J. A. (2000) Updating the average efficiency factor in α-designs. *Submitted for publication.*

Williams, E. R. and Matheson, A. C. (1994) *Experimental Design and Analysis for Use in Tree Improvement.* Melbourne: Commonwealth Scientific and Industrial Research Organisation.

Yates, F. (1936) A new method of arranging variety trials involving a large number of varieties. *J. Agric. Sci.*, **26**, 424–455.