Computer Aided Search of Linear Trend-free Multifactor Designs

संगणक की सहायता से बहुकारक रेखीय प्रवृत्तिमुक्त अभिकल्पनाओं का अन्वेषण

by

Susheel Kumar Sarkar

Thesis submitted to the Faculty of Post-Graduate School,
Indian Agricultural Research Institute, New Delhi,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

AGRICULTURAL STATISTICS

Indian Agricultural Statistics Research Institute
Indian Agricultural Research Institute
New Delhi - 110 012
2008

Computer Aided Search of Linear Trend-free Multifactor Designs

संगणक की सहायता से बहुकारक रेखीय प्रवृत्तिमुक्त अभिकल्पनाओं का अन्वेषण

by

Susheel Kumar Sarkar

Thesis submitted to the Faculty of Post-Graduate School,
Indian Agricultural Research Institute, New Delhi,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
AGRICULTURAL STATISTICS

Indian Agricultural Statistics Research Institute
Indian Agricultural Research Institute
New Delhi - 110 012
2008

Approved By:

Chairman:
Dr. Krishan Lal

Members:

Dr. Rajender Parsad

Prof. V.K. Gupta

Dr. D.R. Singh

Mr. K.K. Chaturvedi

Dr. Krishan Lal
Senior Scientist
I. A. S. R. I., Library Avenue,
Pusa, New Delhi-110 012

CERTIFICATE

This is to certify that the thesis "**COMPUTER AIDED SEARCH OF LINEAR TREND-FREE MULTIFACTOR DESIGNS"** submitted in the partial fulfillment of the requirement for the **DOCTOR OF PHILOSOPHY** in **AGRICULTURAL STATISTICS** of the Faculty of *Post-Graduate School, Indian Agricultural Research Institute, New Delhi*, is a record of bonafide research work carried out by **SUSHEEL KUMAR SARKAR** under my guidance and supervision. No part of the thesis has been submitted for any other degree or diploma.

All assistance and help received during the course of this investigation has been duly acknowledged by him.

23rd February, 2008
New Delhi

(Krishan Lal)
Chairman
Advisory Committee

Acknowledgements

It was the grace of God and my good luck that I got Dr. Krishan Lal, Senior Scientist, Indian Agricultural Statistics Research Institute (I.A.S.R.I.), New Delhi as Chairman of my Advisory Committee. He always gave his innovative ideas, constructive criticism and constant monitoring during the period of this investigation and also in the preparation of this thesis. His affectionate and caring behavior was not only limited to being a guide, he also gave all kind of moral support and help in various stress period of my life. Indeed, if this thesis is going to be submitted, it is due to lot of his efforts.

I express my deep sense of gratitude to Dr. Rajender Parsad, National Fellow, I.A.S.R.I., New Delhi and Co-chairman of my Advisory Committee for his valuable suggestions and punctilious criticism. He worked equally hard with me to give this work a presentable shape. His critical comments and suggestions were helpful to improve my research work. On several occasions when I was less hopeful for research work, his encouragement sparked me. His unlimited help and support is greatly acknowledged.

I am highly grateful to Prof. V.K. Gupta, National Professor, Indian Council Agricultural Research (I.C.A.R.), New Delhi as a member of my Advisory Committee. He has been a source of strength and support through out my study period at IASRI. With reverence, I would like to thank him for many valuable comments, which helped to shape this work. He showed the different ways to approach the problem and the need to be persistent to accomplish any goal.

I am also indebted to Dr. V.K. Sharma, Ex. Head of Division (DoE) and Ex. Professor of Agricultural Statistics for his valuable suggestions, constant help and encouragement throughout the investigation.

I gratefully acknowledge the help of Dr. D.R. Singh, Scientist (Senior Scale), and member of my advisory committee for his useful suggestions during the research work.

I am extremely thankful to Mr. K.K. Chaturvedi, Scientist (Senior Scale), Division of Computer Application, I.A.S.R.I., New Delhi for his useful suggestions during this investigation.

I am also indebted to Dr. V.K. Bhatia, Professor of Agricultural Statistics for his encouragement throughout the investigation. I avail this opportunity to express my profound gratitude to him for providing the necessary research facilities for completion this work.

I am highly grateful to Dr. S.D. Sharma, Director, IASRI for providing all facilities throughout the study.

I am highly grateful to Mr. R.S. Khatri, Principal Scientist and Warden, Sukhatme Hostel, IASRI for providing all kinds of help facilities in the hostel throughout my study period.

I am also thankful to Dr. Prajneshu, Dr. Anil Rai, Dr. R. Srivastava, Dr. P.K. Batra, Dr. L.M. Bhar, Dr. A.K. Paul, Dr. Seema Jaggi, Dr. Tauquer Ahmed, Dr. V. Ramasubrahamanian, Jha Sir, Cinni Madam, for their encouragements.

I take this opportunity to appreciate the help rendered by the staff of TAC, IASRI and staff of CAS Lab. Special thanks are also due to Sondhi ji, Sanjeev ji, Thakur ji, Satveer ji and Raj Madam.

I feel personally obliged to express sincere thanks to Tarak Nath Saha and Swati Khan for constant encouragement and moral support during course and investigation.

I do not have any words to express the help and support given by Raju Sir, Amitava Sir, Popat Sir, Harish Sir, Chandran Sir Sunil Sir, Ballu Sir, Satpati Sir, Sangeeta mam, Pattu Sir, Tomar Sir, Goyal Sir, Guruban Sir, Vinod Sir, Dharmendra Sir, Gullu Sir, Jawaid Sir, Ananta Sir, Ajay Sir, Sumanta Sir, Sandipan Sir, Pradip Sir Dharm Nath, Asif, Sarika, Sanjeev, Ram and Shiv Choudhary, Dwijesh, Patel, Kaushik, Dibyendu, Sandip Uttam, Robin, Maurya, Koushik, Sandip, Pabitra, Soumen, Nilu, Madan, Baidyanath, Ranjit, Meherul, Anesh and all others for their constant help and enthusiasm, which helped me to accomplish this venture.

It gives me great pleasure to express my heartfelt gratitude to Sumit Sharma, Nripendra, Yogeshwar, Khela, Lokesh Babbar, Kaustav, Bishal, Sankalp, Ankur, Alok, Lokesh, Ramji, Sahiram, Manoj, Sukant, Pravin, Ambrish, Sanmit, Niti, Mukesh, Mohan, Arpan, Rakesh, Jay Prakash, Subrata, Sanjay, Vinay, Mahesh, Eldho, Iti, Chyna, Yogita, Sazia, Ragini, Sarita, Snehal and Jayoti.

My Parents did a lot of sacrifices to educate me to this level and to make me what I am today. My affectionate thanks are always due to them for their love, affection, support and constant ushering of blessings on me. I am thankful to my elder brother Santosh and my sisters Mamata for their moral support.

I extend my thanks to the Dean and Director, IARI, New Delhi and staff of PG school for their helpful attitude and cooperation, throughout the period of study. Last but not least I am thankful to the financial assistances that I got from IASRI in forms of SRF during the tenure of my study.

Date: 23rd February, 2008
Place: IASRI, New Delhi-12 (Susheel Kumar Sarkar)

C O N T E N T S

Chapter I

INTRODUCTION AND REVIEW OF LITERATURE

1.1 Introduction

The data generated from designed experiments is used to draw valid inferences about the population. Heterogeneity in the experimental material is the major source of variability to be reckoned within the statistical designing of scientific experiments. Occasionally, one can find a certain factor (called nuisance factor) that though not of interest to the experimenter, does contribute significantly to the variability in the experimental material. Various levels of this factor are used for blocking. In experimental situations with only one nuisance factor, block designs are used. These designs are useful in controlling the heterogeneity of the experimental units and it is ascribed to between blocks variability. But in many experimental situations the response is dependent on the spatial or temporal position of the experimental units within a block and thus trend in the experimental units become another important nuisance factor. In such situations one may think of suitable designs in which treatment effects are orthogonal to trend effects, in the sense that analysis of the design could be done in usual manner, as if no trend effects were present. Such designs may be called trend-free designs.

Such situations may occur in field experiments when there is slope in field or on undulating land in hilly areas. It is because that when the land is irrigated, the nutrients supplied by the fertilizers may not be equally distributed and a slope may cause a trend in experimental units. In the sequel, we describe the experimental situations where such trend may occur over time or space.

Experimental situation 1.1: [Federer and Schlottfeldt (1954)]. **“**An experiment was devised in the spring of 1951, to determine whether the exposure of tobacco seeds to different dosages of cathode rays would affect the growth of the resulting plants. The seeds were from a strain of tobacco that had been under controlled pollination since 1909, and hence, the material used in the experiment was highly uniform with respect to its genetical background. The seven different treatments (the different doses of cathode rays) were laid out in a randomized complete block design with eight replicates. The plot size was 2 rows by 10 plants with 3 feet between rows and 1.5 feet between the plants. The following measurements were made

i) Plant height on 13-07-1951 and 14-08-1951,
ii) Length of longest leaf on 13-07-1951 and 14-08-1951 and
iii) Width of widest leaf on 13-07-1951.

Shortly after the plants were transplanted to the field it became apparent that an environmental gradient existed from the center of the replicates outward. This was confirmed when the data were obtained. The data were analyzed using: a) usual analysis of variance (ANOVA), b) analysis of covariance (ANCOVA) with position of the plots within a replicate as covariates. Upon fitting curvilinear covariance of second degree a considerable reduction in mean square error (MSE) is obtained. In fact, the MSE by ANCOVA was about half that obtained by ANOVA."

Experimental Situation 1.2: [Cox (1958)]. **"**Consider an experiment to investigate the effect on textile process of changing the relative humidity. Suppose that three relative humidities 50, 60 and 70% are to be used. To obtain uniform experimental units a suitable quantity of raw material was taken and thoroughly mixed and then divided into say, nine experimental units. The first batch was processed at one relative humidity in the first period, the second batch at different relative humidity in the second period, and so on. Superimposed on any treatment effects and on random variations remaining, is likely to be a smooth trend due to aging of the material. It would be of interest to estimate this trend explicitly, as well as to set up the experiment so that the trend has little or no influence on the estimates of treatment effects."

Here we have three humidities as three treatments T_{50}, T_{60} and T_{70}. Since the experimental material is having trend over time, so the treatments are to be allocated in the following order to eliminate the effect of linear trend.

Treatments	T_{60}	T_{50}	T_{70}	T_{70}	T_{60}	T_{50}	T_{50}	T_{70}	T_{60}
Plot position	1	2	3	4	5	6	7	8	9
Coefficient of Orthogonal Polynomial (Linear)	-4	-3	-2	-1	0	1	2	3	4

The mean influence of trend on T_{50} is $\frac{1}{3}(2+6+7)=5$, the mean influence of trend on T_{60} is $\frac{1}{3}(1+5+9)=5$ and the mean influence of trend on T_{70} is also $\frac{1}{3}(3+4+8)=5$. Alternate way to examine that allotment of treatments are orthogonal to trend, is to test whether the sum of the coefficients of orthogonal polynomials is zero for each treatment separately. For treatment number T_{50} sum of the coefficients of polynomials (-3+1+2) is zero. Similarly for T_{60} and T_{70} sum of the polynomials is also zero. Hence, treatments allocated are orthogonal to linear trend. Thus any contrast among these treatments is not affected by the linear trend.

Experimental situation 1.3: In an aviary nutrition experiment, four feeds were the four treatments; say T_1, T_2, T_3 and T_4. The chicks were kept in cage-tier system. In a cage there are four tiers one below the other. The experimenter was aware from previous experiences that if chicks are kept from top tier to bottom tier there are chances of trend due to sunlight, fresh air, etc. But the experimenter may not be aware of trend-free designs. The experimenter applied the same treatment to all the tiers in a cage to avoid the effect of any trend. In each tier there were 10 birds. The body weight of the birds from birth to sixth week of age was measured.

To confirm whether there is effect of trend among the tiers within the cage, the obtained data were analyzed by two methods; one by usual analysis of variance (ANOVA) and the other by using analysis of covariance (ANCOVA) with position of tiers as covariates assuming linear trend between the tiers within each cage. The mean square errors of the analysis for live weight from third week of age to sixth week of age by the two methods are given in Table-1.

Table-1.1. ANOVA and ANCOVA Table of live weight of chicks

Week	**ANOVA**			**ANCOVA**		
	Degrees of freedom	**MSE-I**		**Degrees of freedom**	**MSE-I**	**MSE-I/ MSE-II**
III	12	1416.7		11	1173.5	0.8288
IV	12	12902.2		11	12585.7	0.9755
V	12	32571.9		11	28898.9	0.8872
VI	12	51897.9		11	46407.4	0.8942

These results revealed there is a reduction in mean square errors using Analysis of Covariance.

In another aviary experiment, a study was made by Sachdev *et al.* (1989) to investigate the effect of cage-tier system. The study was based on adult female quails, 200 each from line A and B, up to 50 weeks of age and the following observations were made:

1. In Line A, significant effect of cage-tier locations was observed on feed consumption during 6th to 10th and 15th to 18th weeks of age.
2. In Line B, the feed consumption during 23rd and 26th weeks of age, total egg production, feed efficiency and shell weight were significant influenced by cage-tier locations.

A typical type of magic square is given by Philips (1968) as shown in Table1.2. It is a 2^4 factorial experiment, in which A, B, C and D represents the four treatments and their subscripts 0 and 1 are the two levels of each of them. The numbers of the occasions of measurement have been entered in the body of the table as a symmetric magic square. The numbers in the table are the plot positions at which the treatment combinations are to be allocated.

Table 1.2

		C_0		**C_1**	
		D_0	**D_1**	**D_1**	**D_0**
A_0	**B_0**	16 (15)	2 (-13)	3 (-11)	13 (9)
	B_1	5 (-7)	11 (5)	10 (3)	8 (-1)
A_1	**B_1**	9 (1)	7 (-3)	6 (-5)	12 (7)
	B_0	4 (-9)	14 (11)	15 (13)	1 (-15)

Note: Figures in parentheses are the coefficients of orthogonal polynomials.

In above square table of order 4×4, sum of each row, column and diagonal is $\frac{1}{2}n(n^2+1)$ *i. e.* 34. The main effects A, B, C and D are linear trend-free as the average for each of its level is $\frac{68}{8} = 8.5$. For the interaction $\mathrm{A \times B, A \times C, B \times D}$ and $\mathrm{C \times D}$ the mean effect is same for each of its levels *e.g.* in interaction $\mathrm{A \times B}$ for each level of $A_i B_j$, $i, j = 0, 1$ mean effect is $\frac{34}{4} = 8.5$. It shows that these effects are also linear trend-free. But it is not true for interactions $\mathrm{A \times D}$ and $\mathrm{B \times C}$, these two interactions are not linearly trend-free. Also averages of numbers of higher order interactions are not balanced (for example the numbers for third order interactions and $\mathrm{A \times B \times C \times D}$ are inevitably unbalanced).

In these experiments trend-free designs will be more efficient designs. Other experimental situations where trend may occur are Green-house experiments, where source of heat is located on sides of the house; in poultry experiments where the source of heat is at the center of the shed and chicks of early age are in the cages; orchard and vineyard experiments on undulating topography, etc.

1.2 Linear Trend-free Block Designs

The study on systematic designs was initiated by Cox (1951). He considered the arrangement of treatments to experimental units ordered in time or space without blocking and with a smooth polynomial trend is assumed to exist over the entire sequence of experimental units. Other early approaches to the problem of trend elimination were discussed by Box (1952). Box and Hay (1953) gave the method of construction of a certain class of designs with quantity factors by means of which trend occurring during a comparative experiment may be eliminated without loss of efficiency. The design and analysis have been illustrated with an example of bioassay.

Bradley and Yeh (1980) first gave the theory of trend-free block designs. They have considered the situation where a common polynomial trend is assumed to exist over the plots in each block of a classical experimental design. They defined a trend-free block design as a block design in which the adjusted sum of squares due to treatments in a model with the trend effects remains the same as in the model without trend effects. They derived the necessary and sufficient condition for the block design to be trend-free block (TFB) design under the assumption of uniform trend within blocks for a equireplicated and homoscedastic block design model.

Yeh and Bradley (1983) discussed the existence of TFB design for specified trends under a homoscedastic model when each treatment is equally replicated. Some results for linear and odd degree polynomial TFB designs are given by them. Dhall (1986) studied the trend-free incomplete block designs. He prepared a catalogue of trend-free balanced incomplete block (BIB) designs for $3 \leq k \leq 6$ and LTFBIB designs for $7 \leq k \leq 15$. He gave some trend-free group divisible partially balanced incomplete block designs for two-associate class. Bradley and Odeh (1988) gave an algorithm in FORTRAN 77 for the construction of linear trend-free block (LTFB) designs. Stufken (1988) gave a weak point of Yeh and Bradley (1983), " Every binary block design having $r(k+1) \equiv 0$ (mod 2) can be converted into Linear trend-free block design" with an example.

Chai and Majumdar (1993) made a correction to Yeh and Bradley (1983) and proved that a binary block design can be converted into linear trend-free block (LTFB) design when (i) the design is BIB design, or (ii) k, the block size is an even number, or (iii) the design is balanced block design with $b \geq 3$. They also gave a distinct definition of LTFB and strongly linear trend-free block (SLTFB) design. Jacroux, Majumdar and Shah (1995, 1997) developed some methods for identifying efficient designs when different blocks may have linear trend effect of different slope. Majumdar and Martin (2002) extended the above study for quadratic and cubic trend effects. Lal, Parsad and Gupta (2005) obtained the condition for a general block design to be trend-free block design under heteroscedastic model. They prepared the catalogues of binary variance balanced block, balanced incomplete block and partially balanced incomplete block designs that are trend-free. They have also investigated the trend-free nested balanced incomplete block designs and block designs for diallel crosses

In all theses studies, the following usual additive model for a block design with polynomial trend term added is written as

$$y_{jt} = \mu + \sum_{i=1}^{v} \delta_{jt}^{i} \tau_i + \beta_j + \sum_{\alpha=1}^{p} \theta_\alpha \Phi_\alpha (t) + e_{jt} \tag{1.1}$$

for $j = 1, ..., b$; $t = 1, ..., k$, where y_{jt} is the observation on plot t^{th} position of block j; μ, τ_i and β_j are respectively, the usual mean, treatment and block parameters; e_{jt} are random errors assumed to be independently and identically distributed as normal with zero mean and variance σ^2; $\delta_{jt}^{i} = 1 \text{ or } 0$ as the treatment i is or is not on the plot (j, t), $i = 1, . . . , v$. For $\alpha = 1$ the design is said to be linear trend-free block (LTFB) design, for $\alpha = 2$ the design is quadratic trend-free and so on. The maximum value that α can takes is $(k - 1)$. Here we consider the case of $\alpha = 1$ i.e. linear trend free designs only.

A linear trend free-block design can be defined as follows:

Let a design d be represented by a $k \times b$ array of symbols $1, ..., v$, with columns denoting blocks and rows periods, and the class of all connected block designs with v treatments in b blocks each of size k. Thus, if the entry of a cell (i, j) of d is i, it means that under d, treatment i has to be applied in period l of block j and S_{dil} denotes the number of times treatment i appears in row (period) l. Then a design is LTFB design iff

$$\sum_{l=1}^{k} S_{dil} \Phi_1 (l) = 0, \; i = 1, ..., v$$

(1.2)

where Φ_1 (l) is the orthogonal polynomials of degree 1, $l = 1, ..., k$.

1.3 Nearly Linear Trend-free Designs

Sometimes it is not possible to make the design linear trend-free and this provides a motivation to go for nearly linear trend-free designs. Yeh, Bradley and Notz (1985) introduced the concept of nearly linear trend-free block (NLTFB) designs. They defined a NLTFD as a block design under model (1.1) if

$$\sum_{i=1}^{v} \left\{ \sum_{j=1}^{b} \sum_{t=1}^{k} \delta_{jt}^{i} \Phi_1 (t) \right\} \tag{1.3}$$

is minimum among class of connected designs with the same (treatment-block) incidence matrix.

Chai (1995) gave method for construction of nearly trend-free version of balanced incomplete block designs to be NTFB designs. Chai simplified and brought clarity for the condition of nearly linear trend-free block design as

$$(\sum_{l=1}^{k} s_{il}\Phi_1(1))^2 = 1 \text{ for } i = 1,2,\ldots,v.$$

Thus a block design is NLTFB design if

$$\sum_{i=1}^{v}\left[\sum_{l=1}^{l} s_{il}\Phi_1(1)\right]^2 \leq v. \tag{1.4}$$

He also discussed A-, D- and E- optimality of BIB designs for the model that include trend effects.

1.4 Designs for Factorial Experiments

It is known that designs for factorial experiments have been widely used in agricultural, biological and industrial experiments. Similar to block designs, the experimental units in factorial experiment may exhibit a trend over space or time. In factorial experiment, we estimate the main effects, interactions of two factors, three factors and so on. If there is trend in the experimental units, the interest of experimenter is to eliminate the trend effect. It is assumed that most of the trend is reduced, if we are able to eliminate the linear trend. Thus in the presence of linear trend in the experimental units in factorial experiments, the treatments combinations of factorial experiment are allocated to the experimental units such that the contrasts of interest (main effects and lower order interactions) are estimated free from linear-trend. Thus in such experiments the run order of treatment combinations are made so that contrasts of main effects, two factor interaction, three factor interaction, etc. are orthogonal to linear trend.

An ordered application of the treatments to experimental unit over time or space is called a run order. In matrix notations we consider the design for factorial experiment in block design set up with trend terms added as

$$\mathbf{y} = \mathbf{X\beta} + \mathbf{B\gamma} + \mathbf{T}\theta + \mathbf{e}, \qquad (1.5)$$

$$\mathrm{E}(\mathbf{e}) = \mathbf{0}, \mathrm{D}(\mathbf{e}) = \sigma^2\mathbf{I}.$$

The linear-trend vector **T** can be expressed as follows:

$\mathbf{T} = (-(n-1), -(n-3), \ldots, -3, -1, 1, 3, (n-3), (n-1))$, for n even and

$\mathbf{T} = \left(-\frac{n-1}{2}, -\frac{n-1}{2}+1, \ldots, -2, -1, 0, 1, 2, \ldots, \frac{n-1}{2}-1, \frac{n-1}{2}\right)$, for n odd,

and **y** is an $n \times 1$ observable random vector and **X** is the $n \times v$ design matrix, $\mathbf{\beta} = (\beta_1, \ldots, \beta_v)'$, is the vector of parameters of interest, i.e. main effects and interactions which are not aliased with each other nor confounded with blocks, **B** is $n \times b$ design matrix for nuisance parameters and $\mathbf{\gamma} = (\gamma_1, \ldots, \gamma_b)'$, the vector of block effects, **e** is the (n $\times$1) column vector of random errors with zero mean and variance $\sigma^2\mathbf{I}$; $\mathbf{T} = \mathbf{1}_b \otimes \mathbf{t}$, **t** is the linear trend vector equal to that of block size and can be expressed as above . As before, we refer to this linear model as a "conceptual" linear model for factorial experiments with trend. The first k rows of $\mathbf{X}^* = [\mathbf{X} \vdots \mathbf{B} \vdots \mathbf{T}]$ correspond to the k treatment combinations in the first block, the next k rows to the treatment combinations in the second block and so on.

Definition 1.1: (Linear trend-free block designs) A block design modeled by (1.5) is a linear trend-free block design if

$$\mathrm{SS}(\mathbf{X}|\mathbf{B}) = \mathrm{SS}(\mathbf{X} \mid \mathbf{B},\mathbf{T})$$

Using this relation, Bradley and Yeh (1980) obtained a necessary and sufficient condition for a block design modeled by (1.5) be linear trend-free is that

$$\mathbf{X}'\mathbf{T} = \mathbf{0} \qquad (1.6)$$

In simple factorial completely randomized block (CRD) design setup when there is no blocking, the model (1.4) becomes

$$\mathbf{y} = \mathbf{X\beta} + \mathbf{T\theta} + \mathbf{e}, \qquad \mathrm{E}(\mathbf{e}) = \mathbf{0}, \mathrm{D}(\mathbf{e}) = \sigma^2\mathbf{I}. \qquad (1.7)$$

By using the relation SS (**X**) = SS (**X**|**T**) the condition becomes $\mathbf{X}'\mathbf{T} = \mathbf{0}$. (1.8)

Now it is required to mention that the difference between the varietal treatments and factorial experiments in CRD and block design setup in reference to trend-free designs. In varietal treatments **X** is the usual design matrix of treatments verses observations while in factorial experiments **X** is the matrix obtained by writing the columns as the contrasts of main effects and treatment interactions effects. In factorial experiments it is not possible to obtain a design that is linear trend-free for all the treatments effects. The experimenter is always interested to obtain a design that linear trend free for main effects and some of the lower order interactions so that these effects can be estimated more precisely. For example if **A**, **B**, **C**, … are the vector of contrasts of main effects in factorial experiments then matrix of main effects is $\mathbf{X} = [\ \mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \ \ \ldots]$. Thus model (1.6) is applicable when we have complete factorial design in one replication and model (1.5) is applicable when we have more than two blocks in one replication by confounding the higher order interaction.

Similar to block designs, it may not always be possible to make the design for factorial experiment that is linear trend-free for the all effects of interest. This provides a motivation to go for nearly linear trend-free designs for the effects of interest in factorial experiment. Thus condition given by Chai (1995) in (1.4) can be simplified for the factorial experiments. In factorial experiment, let **A** be any contrast of order $n \times 1$ of any effect of interest and **T** be the trend vector (coefficient of orthogonal polynomial) then contrast **A** is linear trend free if

$$\mathbf{A}'\mathbf{T} = 0 \tag{1.9}$$

and nearly linear trend-free if

$$\mathbf{A}'\mathbf{T} \le n, \tag{1.10}$$

where n is the total number of treatment combinations.

Hill (1960) introduced the concept of kept factors. The factors are said to be kept factors if the levels of the factors present or absent in a factorial experiment remains present or absent in the fold-over experiment. Daniel and Wilcoxon (1966) developed plans for sequencing the treatment combinations of two-level full or fractional factorials to experimental units, in order to achieve “better” estimation of specified effects in the presence of trends. Draper and Stoneman (1968) introduced the cost criterion using an eight-run factorial plan. Dickinson (1974) extended the work of Draper and Stoneman (1968) to 2^4 and 2^5 complete factorial plans with the search restricted to minimum cost run orders.

Cheng (1985) gave a theoretical description of the cost structure in two-level factorial designs, and also provided some examples of run orders that satisfied the criteria of trend-freeness and minimum cost. Cheng and Jacroux (1988) further introduced trend-free run orders of two-level designs using group theory and the standard orders of treatment combinations for Yates' algorithm. Coster and Cheng (1988) introduced the generalized foldover scheme (GFS) for generating systematic run orders of any prime number level full and fractional factorial plan.

John (1990) introduced a foldover method developed from the Daniel and Wilcoxon (1966) plans, and discussed the trend free properties of systematic run orders based on their method. Jacrox and SahaRoy (1990) gave some methods of construction of designs where a complete and factorial experiment having all factors at 2 levels using $2^m \times 2^n$ row-coloum design incorporating trend effect. Cheng (1990) discussed the relationship between these two methods given by Cheng and Jacroux (1988), and pointed out that they are actually equivalent. Bailey (1992) gave the method for construction of trend-resistant factorial designs, when one observation can be taken at a time. Coster (1993) extended the work of Coster and Cheng (1988) to mixed level factorial designs. Using the general fold-over scheme (GFS), Coster (1993) provided tables of minimum cost, and linear trend-free run sequences for two- and three-level fractional factorial designs. Jan and Wang (1995) provided algorithms for a run order of two-level designs with minimum cost under the assumption of unequal cost for each factor with or without trend.

1.5 Response Surface Designs

Data from experiments with levels or level combinations of one or more factors as treatments are normally investigated to compare level effects of the factors and their interactions. Though such investigations are useful to have objective assessment of the effect of levels tried in the experiment, this seems to have inadequate in nature. The analysis as such does not give any information regarding the possible effects of the intervening levels of the factors or their combinations, *i.e.* one is not able to interpolate the responses at the treatment combinations not tried in the experiment. In such cases, it is more realistic and informative to carry out investigations to determine and quantify the relationship between the values of one or more measurable response variable (s) and the setting of a group of experimental factors presumed to affect the response (s) and to find the settings of the experimental factors that produce the best value or the best set of values of the response(s).

Thus if all the factors are quantitative in nature, it is proper to think the response as a function of the factor levels and data from quantitative factorial experiments can be used to fit the response surfaces over the region of interest. Response surfaces determine the relationship between the values of one or more measurable response variables and provide the settings of the experimental factors that produce the best values of the response. In addition to it, these provide information about the rate of change of a response variable. They can also indicate the interactions among the quantitative treatment factors.

The special class of designed experiments for fitting the response surfaces is called response surface design. Response surface designs have wide applications in agricultural, biological and industrial experiments. Similar to factorial experiments, experimental units in response surface design may exhibit trend over space or time. Hinkelmann and Jo (1998) first studied linear trend-free response surface designs. They gave a unique solution of equations for k (block size of BIB design) $= 2$ and discussed the procedure for $k \geq 4$ to construct the linear trend free response surface Box- Behnken designs.

1.6 Problem Formulation and Motivation

There are many experimental situations in which the experimental units exhibit a trend. This can be due to the course of sequential application of the treatments to experimental units, the same units or their contributions to the overall response change in a smooth gradient fashion. This may however, be visualized by past experience or previous available data or the similar type of experiments, as explained above. In such situations trend will affect the precision of the estimates of treatment contrasts. It is, therefore, necessary to eliminate the effect of trend by permutation of treatments to the experimental units by using the condition of a design such that treatment effects are estimated free from trend effect. This can be achieved through the use of trend-free designs.

Trend-free designs are quite useful in the experimental situations that may occur in Green house experiments where the source of heat is located on sides of the house and the experimental units (pots) are kept in lines; in poultry experiments where the source of heat is at the centre of the shed and chicks of early age are in the cages; in animal experiments where littermates (animals born in the same litter) are experimental units within a block *i.e.* litters are blocks. Other such experiments are orchard and vineyard experiments on undulating topography, experiments in which response variable of interest is affected by slowly migrating insects entering the area from one side, laboratory experiments where the responses to the experimental units may be affected within time periods by instrument drift or analyst fatigue, etc.

Factorial experiments are being used in agricultural, industrial and biological experiments and trend may occur in the experimental units over time or space. Thus any estimate of contrast for main effects or interaction effects may be affected by this trend effect. In block designs sufficient literature is available on trend-free designs but on factorial experiments not much literature could be traced particularly on trend-free confounded and fractional factorial experiments and response surface designs. In factorial experiments, the interest of the experimenter is to obtain trend-free designs for various treatment effects. Thus it is not so easy to obtain the design for factorial experiments that is trend-free for all or some of the factorial effects. Whatever little is available has not been found much favour from the experimenters.

One of the possible reasons of hampering the applications of these designs is that the construction for such design is not easily available. Therefore, it is required to give easy method of construction, possibly computer aided for the construction of these designs. From the last three decades computer aided search/generation/construction has emerged as a powerful tool to obtain designs for various experimental settings. Various researchers have worked in this field and have developed algorithms/ programs for different situations. This gave a motivation to search/ generate computer aided factorial (complete, confounded, fractional) designs that are at least linear trend-free for main effects. In addition to this, the interest of the experimenter will be to know other treatment combinations that are linear trend free. Some times it is not possible to obtain the effects that are linear trend free. Then it is desired to search whether the effect is nearly linear trend free or not. In many factorial experiments, the factors are quantitative in nature and it is desired to establish a relationship between response and level of various factors. For such situations, response surface designs are used. The run orders in response surface design may also be affected by the presence of trend. Therefore, there is a need to make attempt for computer aided construction of linear trend-free response surface designs. Keeping in view the above the present investigation is taken up with the following objectives:

1.7 Objectives

1. To develop algorithm for computer aided search of linear trend-free designs for two level factorial experiments
2. To develop algorithm for computer aided search of linear trend-free designs for two level fractional factorial experiments
3. To develop algorithm for computer aided search of linear/nearly linear trend-free response surface designs

1.8 Scope of the Present Investigation

The main purpose of the present investigation is to develop computer algorithms for construction of trend-free designs for multifactor experimental settings. The algorithm developed is helpful in generating the factorial experiments that are linear trend-free for main effects. A search has been made to identify the two and three factor interactions that are estimable free from linear trend effect. The developed algorithm is in general and can generate linear trend-free designs for any number of factors each at two levels. The algorithm has also been developed for generation of two level linear trend-free fractional factorial plans. Algorithm has also been developed to generate Box-Behnken response surface designs that are linear trend-free in general and a catalogue is prepared for linear trend-free Box-Behnken response surface designs.

Chapter I of the thesis gives the motivation with some experimental situations for undertaking the problem handled in the present investigation. It is followed by a review of literature, motivation of the problem and objectives of the studies.

Chapter II of the thesis describes the development of algorithm to construct trend-free designs for full factorial experiment when all factors are at two levels. Linear trend-free designs for confounded factorial experiments have also been obtained. The algorithm along with the computer program has been given in general for any number of factors each at two levels for complete factorial in a replication and confounded factorial experiments separately. The catalogues of complete and confounded factorial experiments is given that is restricted to factorial experiments for $k = 3$ to 7 factors each at 2 levels for full factorial in a replication and confounded factorial experiments separately has been prepared in this chapter.

Chapter III is devoted to obtain trend-free fractional factorial designs. In this chapter using the equation solving, trend-free fractional factorial designs have been developed. For fractional factorial experiments the algorithm along with the computer program has been given in general for any number of factors each at two levels. The catalogue has been prepared for $k = 5$ to 8 factors each at 2 levels.

Chapter IV is devoted to obtain computer aided linear trend-free response surface designs. In this chapter algorithm is developed to generate Box-Behnken response surface designs with the help of a given BIB design. Further algorithm is developed to search linear trend-free Box-Behnken from the obtained designs. The procedure is developed in general and the catalogue for linear trend-free Box-Behnken response surface designs is given for *k* taking values as 2, 3, 4 and 5.

In the last, summary of the present research work are given. The thesis is concluded with a list of references.

Chapter II

DEVELOPMENT OF LINEAR TREND-FREE DESIGNS FOR TWO LEVEL FACTORIAL EXPERIMENTS

2.1 Introduction

Experiments in which the effects (main effects and interactions) of more than one factor are studied together are called factorial experiments. In general, if there are k factors, say, $F_1, F_2, \ldots, F_k$ and the i^{th} factor has s_i levels, $i = 1, \ldots, k$, then the total number of treatment combinations are $\prod_{i=1}^{k} s_i$. Factorial experiments are of two types.

1. **Symmetric Factorial Experiments**: In these experiments the number of levels of all factors is same *i.e.* $s_i = s \;\; \forall \; i = 1, \ldots, k$.
2. **Asymmetrical Factorial Experiments**: In these experiments the number of levels of all factors is not same i.e. there are at least two factors for which the number of levels is not same. These are also known as mixed level factorial experiments.

In factorial experiments, 2^k is a symmetrical factorial experiment with k factors each at 2 levels. In it there are $2^k - 1$ factorial effects (main effects and interaction effects) and contrasts of all these treatment combinations are orthogonal to each other. 2^k factorial experiment is known as the simplest type of factorial experiment. The two levels of a factor may be its presence or absence or a high and a low dose or two modes of application of a technique. In factorial experiments, the experimenter is interested to estimate all the main effects and interactions. This can be achieved through designing complete factorial experiments conducted by using a complete randomized or randomized complete block design with more than one replication.

The factorial experiments have many advantages over single factor experiments as these experiments provide an opportunity to study not only the individual effects but also their interactions. These experiments further economize the experimental resources. When the experiments are conducted factor by factor a large number of experimental units are required for getting the same precision of estimation as one would have got when all the factors are experimented together in the same experiment. There is thus considerable amount of saving of the resources.

When the number of factors and/ or levels increases, the number of treatment combinations becomes so large that it is not possible to accommodate them without losing homogeneity within block. To handle such situations, the concept of confounding is introduced in factorial experiments. In confounded factorial experiments, there are more than one block per replication and we lose information on the factorial effects whose contrasts become identical to the block contrasts in a given replication.

In factorial experiments, the interest of the experimenter is to obtain trend-free designs for various treatment effects when the experimental units are having trend-effect over time or space. Sufficient literature is available on linear trend-free factorial experiments. But not much is available on linear trend-free factorial experiments simultaneously on main effects and lower order interactions and linear trend-free confounded factorial experiments. Moreover the construction of linear trend-free factorial experiments available in the literature is algebraically involved. Therefore, it is required to give easy method of construction, possibly computer aided for the construction of these designs.

Some of the studies on computer aided designs are available in the literature in different areas. Nguyen (1983) gave algorithms for construction of D-optimal fractional factorial plans and MS-optimal incomplete block designs. Bradley and Odeh (1988) gave an algorithm in FORTRAN 77 for the construction of linear trend-free block (LTFB) designs. Dwivedi (1997) developed the algorithm for search of optimal block designs, minimal connected D-optimal designs and D-optimal saturated main effect plans. Rathore (2004), Rathore *et al.* (2004) developed algorithms for computer aided search of optimal/ nearly optimal balanced treatment incomplete block designs, designs for making treatment-treatment and treatment-control comparisons. Satpati (2006), Satpati *et al.*(2006) investigated computer aided search of efficient designs for dependent observations, correlated error structure for nested block designs, block designs and designs for making test treatment control treatment comparisons and change over designs. Kohli (2006) carried out computer aided construction of two level supersaturated designs. No work seems to be available on computer aided construction on trend-free factorial experiments. This chapter, therefore, deals with the computer aided construction of complete factorial experiments without confounding and with confounding that are linear trend-free for main effects and identification of two and three factor interactions that are linear trend-free/ nearly linear trend-free in the obtained designs.

Section 2.2 deals with general description of linear trend-free factorial experiments each factor at two levels. Under Section 2.3 algorithm to obtain the two level factorial experiments for any number factors has been described for generation of linear trend-free two level factorial experiments in which main effects are estimated free from linear trend effects. This algorithm is extended to make a search for identification of two and three factor interactions that are estimable free from linear trend effects. In Section 2.4 working of the algorithm for complete factorial to be linear trend-free for main effects has been illustrated. In Section 2.5, algorithm has been developed to generate confounded factorial experiments that are linear trend-free for main effects and search has been made to investigate the two and three factor interactions that are estimable free from trend. In Section 2.6 working of algorithm for confounded factorial experiments has been described.

The algorithms developed have been translated in computer program in Microsoft Visual C++ and the program is given in Appendix 2.1. The catalogues of the 2^k factorial experiments for (k = 3, …, 7) that are linear trend-free for main effects are given in Appendix 2.2. The identified two and three factor interactions that are linear trend-free are given along with the design.

2.2 Linear Trend-free Designs for Factorial Experiments

Designs for factorial experiments have been widely used in agricultural, biological and industrial experiments. Similar to block designs, the experimental units in factorial experiment may exhibit trend over space or time. In factorial experiments, the interest of the experimenter is to estimate the lower order factorial effects precisely. Thus in the presence of the trend in experimental units, a search is made to identify contrasts of two and three factor interactions that are linear/nearly linear trend-free.

For a design of 2^k factorial experiment the total number of treatment combinations are $n = 2^k$ and the model for factorial experiment with linear trend effect is

$$\mathbf{y} = \beta_0\mathbf{1} + \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \cdots + \mathbf{X}_k\beta_k + \mathbf{T}\theta + \mathbf{e}, \qquad (2.1)$$

$$\mathrm{E}(\mathbf{e}) = \mathbf{0}, \mathrm{D}(\mathbf{e}) = \sigma^2\mathbf{I} \ ,$$

where **y** is a n × 1 vector of observations,

1 is the vector of 1's,

$\mathbf{X}_1$ is a matrix of contrasts of order $n \times k$ of k main effects,

$\mathbf{X}_2$ is the matrix of contrasts of order $n \times \binom{k}{2}$ of $\binom{k}{2}$ two factor interactions, and so on,

$\mathbf{X}_k$ is a vector of contrast of order $n \times 1$ of a k factor interaction,

β_0 is a constant,

$\boldsymbol{\beta}_1$ is the $k \times 1$ vector of a coefficient for main effects,

$\boldsymbol{\beta}_2$ is the $\binom{k}{2} \times 1$ vector of a coefficient for two factor interactions and so on,

β_k is the coefficient of k-factor interaction,

T is the vector of coefficient of first degree polynomial of order *n*,

θ is the regression coefficient and

e is $n \times 1$ vector of identically independently normally distributed errors.

In 2^k factorial experiment, total number of treatment combination is even. So vector **T** can be defined as

$$\mathbf{T} = \left(-(\mathrm{n}-1), -(\mathrm{n}-3), \ldots, -3, -1, 1, 3, (\mathrm{n}-3), (\mathrm{n}-1)\right) \tag{2.2}$$

Here our interest is to obtain designs for factorial experiments in which contrasts for the main effects are estimated free from linear trend effects and to identify/search two and three factor interactions that are estimable free from linear trend effect. To obtain such designs we begin with following Lemmas:

Lemma 2.1: In any run order of a complete 2^k design, the number of mutually orthogonal linear trend-free factorial contrasts (main effects or interaction) is at most $2^k - k - 1$ (Cheng and Jacroux, 1988).

Let $s_1 \circ s_2 \circ \ldots \circ s_k$ be the vector of coefficients of contrast of first, second and so on k^{th} factor and $\circ$ represents the component-wise product of two vectors. Then we have the following Lemma:

Lemma 2.2: Let $s_1 \circ s_2 \circ \ldots \circ s_k$ is the contrast of main effect, then it is orthogonal to $(\text{k-1})^{\text{th}}$ order trend (Cheng and Jacroux, 1988).

The algorithm to generate the design is given in Section 2.3.

2.3 Algorithm for linear trend-free factorial experiment

The algorithm for obtaining designs for complete factorial experiments without confounding which are trend-free for estimation of main effects and to identify the 2- and 3- factor interactions that are estimable free from linear trend effect is developed in three steps which are described in the sequel.

I. Algorithm to generate the two level factorial experiments

I.1 Let us have k factors.

I.2 The number of treatment combinations is $n = 2^k$. Here n is an even number.

I.3 Make an array of dimension $n \times k$.

I.4 The first column of the array of size n is made such that the entries are -1 and

+1 alternatively.

I.5 The second column of the array is made such that the entries are in combinations of two i.e. -1, -1 and +1, +1 alternatively, and so on.

I.6 The k^{th} column is made such that the first n/2 places are filled with -1 and last n/2 places with +1.

I.7 In the obtained $n \times k$ matrix, each column is the coefficient of contrast of main effects separately and thus *k* columns are coefficients of contrast of *k* factors.

I.8 The n rows of $n \times k$ matrix represent $n\ (= 2^k)$ treatment combinations in lexicographic order.

The obtained factorial experiment in lexicographic order has been converted into linear trend free for main effects by using the algorithm given below:

II. Algorithm to generate linear trend-free two level factorial experiments

From the generated 2^k factorial experiment above, we proceed as follows:

II.1 In 2^k factorial experiment, the treatment combinations are in standard order.

II.2 Let $s_1, s_2, \ldots, s_k$ are the *k* vectors denoting the coefficients of contrast for main effects that are in standard order i.e.

$$\begin{array}{cccc}
s_1 & s_2 & \cdots & s_k \\
-1 & -1 & & -1 \\
1 & -1 & & -1 \\
-1 & 1 & & -1 \\
1 & 1 & & -1 \\
\vdots & \vdots & \cdots & \vdots \\
-1 & -1 & & 1 \\
1 & -1 & & 1 \\
-1 & 1 & & 1 \\
1 & 1 & & 1
\end{array}$$

II.3 Component-wise product of s_i and s_j $(s_i \circ s_j)$ is made to develop the desired designs for factorial experiments.

II.4 There are two methods for k odd and k even.

For *k* is odd: When k is odd the new coefficient vectors for main effects is obtained as:

Let A_i be the new coefficient vectors for main effects for i^{th} factor, $i = 1, 2, \ldots, k$

$A_i = s_1 \circ s_2 \circ \cdots \circ s_{i-1} \circ s_{i+1} \circ \cdots \circ s_k$; $(i = 1, 2, \ldots, k-1)$

and $A_k = s_1 \circ s_2 \circ \cdots \circ s_k$; $(i = k)$

For *k* is even: When k is even the new coefficient vectors for main effects is obtained as:

$A_i = s_1 \circ s_2 \circ \cdots \circ s_{i-1} \circ s_{i+1} \circ \cdots \circ s_k$; $(i = 1, 2, \ldots, k)$.

II.5 Thus k columns corresponding to k main effects are generated.

II.6 A trend vector (**T**) is generated using (2.2).

II.7 To examine whether obtained columns (coefficients of main effect) are linear trend-free, multiply each column with trend vector and sum it.

II.8 Sum of all product columns is made. It is checked that sum of all product column is zero or not. The column (main effect) for which the sum is zero, is linear trend-free otherwise not.

III. Algorithm to search two and three factors interactions that are linear trend-free

From the design obtained in first two steps, a search has been made for two and three factor interactions that are linear or nearly linear trend-free.

For two factor interactions

III.1 From the obtained k columns (coefficients of main effects) in II.5, we multiply each pair of columns in all possible ways.

III.2 ${}^kC_2$ columns are obtained, that are the coefficients of contrasts of two factor interactions.

III.3 Multiply each of the above columns with trend vector.

III.4 The column (coefficient of two factor interaction) for which the product sum is zero, is linear trend-free.

III.5 The columns for which product sum is $\leq n$ (total number of treatment combinations), are nearly linear trend-free (as discussed in (1.10)) and others are not.

For three factor interactions

III.6 From the obtained k columns in II.5, we multiply columns in combinations of three in all possible ways

III.7 Thus ${}^{k}C_{3}$ columns are obtained, that are the coefficients of contrasts of three factor interactions.

III.8 Multiply each of the above columns with trend vector.

III.9 The column (coefficient of three factor interaction) for which the sum is zero, is linear trend-free.

III.10 The columns for which product sum is ≤ n, are nearly linear trend-free and others are not.

The part-III of the algorithm gives two and three factor interactions that are estimable linear trend-free/nearly linear trend-free.

The above algorithm was translated in Microsoft Visual C++ program. The listing of the program is given in Appendix 2.1. From this program, one can obtain the desired factorial experiment for any number of factors k (≥ 3). The catalogue of the obtained design for 2^k factorial experiment (for $k = 3, \ldots, 7$) that are linear trend-free for main effects is given in Appendix 2.2. The identified two and three factor interactions that are linear trend-free are given along with the design.

All the above three parts of the algorithm have been illustrated with the help of an example in Section 2.4.

2.4 Working of algorithm

Consider the problem of constructing a 2^4 factorial experiment with four factors A, B, C and D. In which all four main effects linear trend-free.

Using the computer algorithm described in Section 2.3, we first obtain 16×4 array by using the steps I.3 to I.6. This gives coefficients of four main effects in standard order. These are the main effects of 2^4 factorial design in lexicographic order. We denote the contrasts of main effects as s_a, s_b, s_c and s_d in Design-2.1.

Design-2.1

Treatment combinations	s_a	s_b	s_c	s_d
(1)	-1	-1	-1	-1
a	1	-1	-1	-1
b	-1	1	-1	-1
ab	1	1	-1	-1
c	-1	-1	1	-1
ac	1	-1	1	-1
bc	-1	1	1	-1
abc	1	1	1	-1
d	-1	-1	-1	1
ad	1	-1	-1	1
bd	-1	1	-1	1
abd	1	1	-1	1
cd	-1	-1	1	1
acd	1	-1	1	1
bcd	-1	1	1	1
abcd	1	1	1	1

Using computer program of algorithm-II, perform the component-wise product of s_i and s_j $(s_i \circ s_j)$ by using step II.4. Here *k* (= 4) is even so factor A has been generated using component-wise product of s_b, s_c and s_d as: $\mathrm{A} = \mathrm{s_b} \circ \mathrm{s_c} \circ \mathrm{s_d}$.

s_b	s_c	s_d	$s_b \circ s_c \circ s_d$		A
-1	-1	-1	-1*-1*-1	=	-1
-1	-1	-1	-1*-1*-1	=	-1
1	-1	-1	1*-1*-1	=	1
1	-1	-1	1*-1*-1	=	1
-1	1	-1	-1*1*-1	=	1
-1	1	-1	-1*1*-1	=	1
1	1	-1	1*1*-1	=	-1
1	1	-1	1*1*-1	=	-1
-1	-1	1	-1*-1*1	=	1
-1	-1	1	-1*-1*1	=	1
1	-1	1	1*-1*1	=	-1

1	-1	1	1*-1*1	=	-1
-1	1	1	-1*1*1	=	-1
-1	1	1	-1*1*1	=	-1
1	1	1	1*1*1	=	1
1	1	1	1*1*1	=	1

Similarly B, C and D are generated using the formula:

$A_i = s_1 \circ s_2 \circ \cdots \circ s_{i-1} \circ s_{i+1} \circ \cdots \circ s_k$; $(i = 1, 2, \ldots, k)$.

Thus we get the Design-2.2

Design-2.2

Treatment combinations	**A**	**B**	**C**	**D**
(1)	-1	-1	-1	-1
bcd	-1	1	1	1
acd	1	-1	1	1
ab	1	1	-1	-1
abd	1	1	-1	1
ac	1	-1	1	-1
bc	-1	1	1	-1
d	-1	-1	-1	1
abc	1	1	1	-1
ad	1	-1	-1	1
bd	-1	1	-1	1
c	-1	-1	1	-1
cd	-1	-1	1	1
b	-1	1	-1	-1
a	1	-1	-1	-1
abcd	1	1	1	1

A trend vector ($\mathbf{T}'$) is generated using (2.2) i.e. the coefficients of linear orthogonal polynomial are generated

$\mathbf{T}' =$ -15 -13 -11 -9 -7 -5 -3 -1 1 3 5 7 9 11 13 15

To examine whether obtained columns (coefficients of main effect) of the Design-2.2 are linear trend-free, we multiply each column of Design 2.2 with trend vector $\mathbf{T}'$ and sum it. This is presented in Table 2.1.

Table 2.1

Treatment	**A**	**B**	**C**	**D**

combinations				
(1)	15	15	15	15
bcd	13	-13	-13	-13
acd	-11	11	-11	-11
ab	-9	-9	9	9
abd	-7	-7	7	-7
ac	-5	5	-5	5
bc	3	-3	-3	3
d	1	1	1	-1
abc	1	1	1	-1
ad	3	-3	-3	3
bd	-5	5	-5	5
c	-7	-7	7	-7
cd	-9	-9	9	9
b	-11	11	-11	-11
a	13	-13	-13	-13
abcd	15	15	15	15
Sum	**0**	**0**	**0**	**0**

Table 2.1 shows that all column sums are zero. Hence Design-2.2 is linear trend-free for main effects. Here the meaning of Design-2.2 is linear trend-free for main effects is that if we allocate the 16 treatments combinations to 16 experimental units in the order as in Design- 2.2 then the contrasts of main effects are estimated orthogonal to linear trend and thus these contrasts are said to be linear trend-free.

In Design-2.2 not only main effects are linear trend-free but some other treatment combinations may also be linear trend-free. Since the interest of the experimenter is to estimate low order interaction more precisely so we shall search the treatment combinations of two and three factor interactions that are linear trend-free in obtained Design-2.2 by using algorithm-III.

From the obtained 4 columns (coefficients of main effects) in Design-2.2, we multiply each pair of column in all possible ways by using step III.1 and III.2. Here ${}^4C_2 = 6$ columns are obtained and these are the 6 coefficients of two factor interaction given in Table 2.2.

Table - 2.2

AB	**AC**	**AD**	**BC**	**BD**	**CD**

1	1	1	1	1	1
-1	-1	-1	1	1	1
-1	1	1	-1	-1	1
1	-1	-1	-1	-1	1
1	-1	1	-1	1	-1
-1	1	-1	-1	1	-1
-1	-1	1	1	-1	-1
1	1	-1	1	-1	-1
1	1	-1	1	-1	-1
-1	-1	1	1	-1	-1
-1	1	-1	-1	1	-1
1	-1	1	-1	1	-1
1	-1	-1	-1	-1	1
-1	1	1	-1	-1	1
-1	-1	-1	1	1	1
1	1	1	1	1	1

Multiply each of the columns of Table 2.2 with trend vector $\mathbf{T}'$ using step III.3. The obtained columns and their columns (coefficient of two factor interaction) sum are given in Table 2.3.

Table-2.3

	AB	**AC**	**AD**	**BC**	**BD**	**CD**
	-15	-15	-15	-15	-15	-15
	13	13	13	-13	-13	-13
	11	-11	-11	11	11	-11
	-9	9	9	9	9	-9
	-7	7	-7	7	-7	7
	5	-5	5	5	-5	5
	3	3	-3	-3	3	3
	-1	-1	1	-1	1	1
	1	1	-1	1	-1	-1
	-3	-3	3	3	-3	-3
	-5	5	-5	-5	5	-5
	7	-7	7	-7	7	-7
	9	-9	-9	-9	-9	9
	-11	11	11	-11	-11	11
	-13	-13	-13	13	13	13
	15	15	15	15	15	15
Sum	**0**	**0**	**0**	**0**	**0**	**0**

Now using the condition given in 1.9 and 1.10, treatment combinations for which the product sum are zero, is linear trend-free and the columns for which product sum is

≤ 16, are nearly linear trend-free and others are not nearly linear trend-free. Since all the column sums are zero; hence Design-2.2 is linear trend-free for all the six two factor interactions.

Now to examine three factor interactions, we proceed in as below:

For three factor interactions we multiply columns in combinations of three in all possible ways i e. ${}^4C_3 = 4$ columns are obtained. These columns are the coefficients of three factors interactions and are presented in Table-2.4.

Table-2.4

ABC	**ABD**	**ACD**	**BCD**
-1	-1	-1	-1
-1	-1	-1	1
-1	-1	1	-1
-1	-1	1	1
-1	1	-1	-1
-1	1	-1	1
-1	1	1	-1
-1	1	1	1
1	-1	-1	-1
1	-1	-1	1
1	-1	1	-1
1	-1	1	1
1	1	-1	-1
1	1	-1	1
1	1	1	-1
1	1	1	1

Similar to two factor interaction, multiply each of the three factor interaction obtained in Table 2.3 with trend vector $\mathbf{T}'$ and sum these products. These sums are given Table-2.5.

Table-2.5

ABC	**ABD**	**ACD**	**BCD**
15	15	15	15
13	13	13	-13
11	11	-11	11
9	9	-9	-9
7	-7	7	7
5	-5	5	-5

	3	-3	-3	3
	1	-1	-1	-1
	1	-1	-1	-1
	3	-3	-3	3
	5	-5	5	-5
	7	-7	7	7
	9	9	-9	-9
	11	11	-11	11
	13	13	13	-13
	15	15	15	15
Sum	**128**	**64**	**32**	**16**

We see that among three factor interactions BCD has column total 16 i.e. $\leq n$. Hence using condition (1.10) only treatment combination BCD is nearly linear trend-free and the other 3 three-factor interactions i e. ABC, ABD and ACD are neither linear nor nearly linear trend-free.

Remark 2.1: Here we have illustrated the construction of a 2^4 factorial experiment with factors A, B, C and D. If we permute the treatments, we can obtain 4! factorial experiments that all are linear trend-free for main effects. But the treatment combinations that are linear trend-free of order two and three may change and the number may vary. This has been illustrated by taking 2^3 factorial experiment in the Catalogue given in Appendix 2.2. In the catalogue, first design 2^3 is listed and its five more (total 6) designs that are linear trend-free for main effects are given and the interactions that are linear trend-free or nearly linear trend-free are given below the design .

Further in factorial experiment as the number of factors and/or levels increase, the number of treatment combinations becomes so large that it is not possible to accommodate them without losing homogeneity within block. To handle such situations the concept of confounding is introduced in factorial experiments. In a confounded factorial experiment, there is more than one block per replication and information on the factorial effects whose contrast becomes identical to the block contrast is lost. In confounded factorial experiments, there may be trend effect within blocks. Thus there is need to have confounded factorial designs that eliminates the effect of linear trend.

In 2^k factorial experiment, let there be p independent treatment combinations that are confounded. Then we have $2^p = b$ blocks each of size $2^{k-p} = m$, say. For obtaining the confounded factorial experiment that is linear trend-free for main effects, we use the model and the condition given in (1.5) and (1.6). The condition is

$$\mathbf{X}'\mathbf{T} = \mathbf{0}$$

where $\mathbf{X}$ = [$\mathbf{A}$ $\mathbf{B}$ $\mathbf{C}$ …] and **A**, **B**, **C**, … are the vector of contrasts of main effects in factorial experiments and the vector $\mathbf{T} = \mathbf{1}_b \otimes \mathbf{t}$, $\mathbf{t}$ is the (m × 1) linear trend vector for block size m and b is the number of blocks. The matrix **X** is partitioned into b blocks as $\mathbf{X} = [\mathbf{X}_1' \quad \mathbf{X}_2' \cdots \mathbf{X}_b']'$. Each $\mathbf{X}_i$, i = 1, 2, …, b is having the vector of contrasts of main effects **A**, **B**, **C**, … such that each factor is at its high level and low level equal times, so that main effects are estimable in each block and are not confounded with blocks. With this the algorithm is developed for confounded factorial experiments that are at least linear trend-free for main effects and for some of the two and three factor interactions and is given in Section 2.5.

2.5 Algorithm for linear trend-free designs in confounded factorial experiments

To obtain the desired design, first generate the 2^k confounded factorial experiment in which higher order interactions are confounded with block.

I. Algorithm for confounded factorial experiment

I.1 Let there are k factors, each at two levels.

I.2 Total number of treatment combinations is $n = 2^k$.

I.3 Generate full factorial design in standard (lexicographic) order using the steps I.3 to I.8 of the algorithm given in Section 2.3.

I.4 For solving the equation, change all -1 entries into 0.

I.5 First decide the treatment effect that has to be confounded.

I.6 Solve the following equations for the chosen contrast (treatment combination)

$$\sum x_j = 0$$
$$\sum x_j = 1; (j = 1, 2, \ldots, k)$$

where j takes the values of chosen contrast for the treatment combination to be confounded $(j = 1, 2, \ldots, k)$.

I.7 Similarly other equations are solved if more than one-treatment combinations are to be confounded by following the principle of confounding.

I.8 If p independent treatment combinations are confound, then this algorithm generates 2^p blocks each of size 2^{k-p}.

To rearrange the treatment combination within each block, such that the design is linear trend-free for main effects, use the following steps:

II. Algorithm to generate linear trend-free two level factorial experiments

II.1 The treatment combinations within these blocks are maintaining the same order of sequence as in the lexicographic order of complete factorial experiment.

II.2 Change all 0 entries into -1.

II.3 Here we have $b = 2^p$ blocks each of size $m = 2^{k-p}$.

II.4 Let s_i, i = 1, 2, …, k represent the coefficient of the contrasts of the k main effects. Perform component-wise product within each of b blocks separately. Similar to complete factorial experiment, there are two procedures of component-wise product. The procedure of component-wise product is as given below:

For *k* is odd: When k is odd the new coefficient vectors for main effects will be obtained as:

Let A_i be the new coefficient vectors for main effects for ith factor, $i = 1, 2, \ldots, k$

$$A_i = s_1 \circ s_2 \circ \cdots \circ s_{i-1} \circ s_{i+1} \circ \cdots \circ s_k\, ;\ (i = 1, 2, \ldots, k-1)$$

and $A_k = s_1 \circ s_2 \circ \cdots \circ s_k\, ;\ (i = k)$

For *k* is even: When k is even the new coefficient vectors for main effects will be obtained as:

$$A_i = s_1 \circ s_2 \circ \cdots \circ s_{i-1} \circ s_{i+1} \circ \cdots \circ s_k\, ;\ (i = 1, 2, \ldots, k).$$

II.5 These b blocks are kept in same order as in the complete design.

II.6 Now we have $\mathbf{X}_1 \quad \mathbf{X}_2 \ldots \mathbf{X}_b$ be b matrices of main effects of b blocks.

II.7 Make the matrix **X** by arranging $\mathbf{X}_1 \quad \mathbf{X}_2 \ldots \mathbf{X}_b$ as $\mathbf{X} = [\mathbf{X}_1' \quad \mathbf{X}_2' \cdots \mathbf{X}_b']'$ and the trend vector **T** as $\mathbf{T} = \mathbf{1}_b \otimes \mathbf{t}$, where vector **t** is trend vector of order m.

II.8 The condition $\mathbf{X}'\mathbf{T} = \mathbf{0}$ is checked whether the main effects are linear trend-free.

III. Algorithm to search two and three factors interactions that are linear trend-free.

For two factor interaction

III.1 From the obtained k columns (main effects) of matrix **X**, we multiply each pair of column within each block.

III.2 ${}^kC_2$ columns are obtained, that are the coefficients of two factor interactions.

III.3 Multiply each of the coefficients of two factor interactions of matrix **X** with trend vector **T.**

III.4 The columns (coefficients of two factor interaction) for which the sum is zero, is linear trend-free.

III.5 The coefficient of two factor interaction for which the sum is $\le n$ is nearly linear trend-free.

For three factor interaction

III.6 From the obtained k columns (main effects) of matrix **X**, we multiply each combination of three columns in all possible ways.

III.7 ${}^kC_3$ columns are obtained, that are contrasts of three factor interactions of matrix **X** with trend vector **T.**

III.8 The columns (coefficients of three factor interaction) for which the sum is zero, are linear trend-free.

III.9 The coefficients of three factor interaction for which the sum is $\le n$ are nearly linear trend-free.

The algorithms given in Section 2.3 generate/ search the factorial experiments that are at least linear trend-free for main effects when all the treatment combinations of a complete factorial are to be allocated to experimental units in one replication and the experimental units are having the trend effect. Sometimes it is not possible to accommodate all the treatment combinations in one block because of large number of treatment combinations in one replication then one has to go for having the homogenous blocks of smaller size in one replication by confounding the higher order interactions. In such situations uniform trend may occur within each block and thus in Section 2.5 algorithms have been developed that generate/ search the confounded factorial experiments in which the contrasts of main effects are estimated at least free from linear trend and further identifies the 2- and 3- factor interactions whose contrasts are free from linear trend.

Working of the algorithms given in Section 2.5 has been elaborated by developing a 2^4 confounding factorial experiment by confounding higher order interaction that is at linear trend-free for main effects and identifies 2- and 3- factor interaction whose contrasts are free from linear effect.

2.6 Working of the algorithm

Using algorithm I of section 2.3, with four factors each at two levels, the full factorial experiment in lexicographic order generated and is same as the Design 2.1 in Section 2.4. Using step I.4 of section 2.5, change all -1 into 0 and the design is

Design-2.3

(1)	0	0	0	0
a	1	0	0	0
b	0	1	0	0
ab	1	1	0	0
c	0	0	1	0
ac	1	0	1	0
bc	0	1	1	0
abc	1	1	1	0
d	0	0	0	1
ad	1	0	0	1

bd	0	1	0	1
abd	1	1	0	1
cd	0	0	1	1
acd	1	0	1	1
bcd	0	1	1	1
abcd	1	1	1	1

In the above design factorial effect ABCD has to be confounded. Then using step I.6 of section 2.5, two blocks are obtained by solving following two equations:

$$s_a + s_b + s_c + s_d = 0$$
$$s_a + s_b + s_c + s_d = 1$$

Now change all 0 into -1 using step II.2 of section 2.5. The two blocks of the design are:

Design 2.4

Block – 1

	A	B	C	D
(1)	-1	-1	-1	-1
ab	1	1	-1	-1
ac	1	-1	1	-1
bc	-1	1	1	-1
ad	1	-1	-1	1
bd	-1	1	-1	1
cd	-1	-1	1	1
abcd	1	1	1	1

Block – 2

	A	B	C	D
a	1	-1	-1	-1
b	-1	1	-1	-1
c	-1	-1	1	-1
abc	1	1	1	-1
d	-1	-1	-1	1
abd	1	1	-1	1
acd	1	-1	1	1
bcd	-1	1	1	1

Perform component-wise product separately within each block, using step II.4 of section 2.5. The obtained design is

Design 2.5

Block – 1

	A	B	C	D
(1)	-1	-1	-1	-1

Block – 2

	A	B	C	D
bcd	-1	1	1	1

ab	1	1	-1	-1
ac	1	-1	1	-1
bc	-1	1	1	-1
ad	1	-1	-1	1
bd	-1	1	-1	1
cd	-1	-1	1	1
abcd	1	1	1	1
acd	1	-1	1	1
abd	1	1	-1	1
d	-1	-1	-1	1
abc	1	1	1	-1
c	-1	-1	1	-1
b	-1	1	-1	-1
a	1	-1	-1	-1

To test whether the obtained design is linear trend-free for main effects, use step II.7 and II.8.

A	B	C	D				A	B	C	D
-1	-1	-1	-1	-7			7	7	7	7
1	1	-1	-1	-5			-5	-5	5	5
1	-1	1	-1	-3			-3	3	-3	3
-1	1	1	-1	-1			1	-1	-1	1
1	-1	-1	1	1			1	-1	-1	1
-1	1	-1	1	3			-3	3	-3	3
-1	-1	1	1	5			-5	-5	5	5
1	1	1	1	7			7	7	7	7
-1	1	1	1	-7	→		7	-7	-7	-7
1	-1	1	1	-5			-5	5	-5	-5
1	1	-1	1	-3			-3	-3	3	-3
-1	-1	-1	1	-1			1	1	1	-1
1	1	1	-1	1			1	1	1	-1
-1	-1	1	-1	3			-3	-3	3	-3
-1	1	-1	-1	5			-5	5	-5	-5
1	-1	-1	-1	7			7	-7	-7	-7
						Sum	**0**	**0**	**0**	**0**

Thus obtained Design 2.5 is linear trend-free for main effects. We further investigate two and three factor interactions that are linear trend-free in Design 2.5 using algorithm-III of section 2.5. Using step III.2 six two factor interactions are obtained. Further multiply each of the coefficients of two factor interactions with trend vector. From step III.4 it is found that all the two factor interactions i.e. AB, AC, AD, BC, BD and CD are linear trend-free. Further three factor interactions are investigated using algorithm steps III.6 to III.9 of section 2.5. Among three factor interactions BCD is linear trend-free, ACD is nearly linear trend-free and the other two ABC and ABD are neither linear trend-free nor nearly linear trend-free.

By using the above algorithm given in section 2.5, program has been written using Microsoft Visual C++. The listing of the program is given in Appendix 2.3. From this program, we can obtain the desired confounded factorial experiment in more than one block per replication by confounding higher order interaction for any number of factors k (≥ 3). The catalogue of the obtained design for 2^k confounded factorial experiment (for $k = 3, \ldots, 7$) that are linear trend-free for main effects is given in Appendix 2.4. The identified two and three factor interactions that are linear trend-free are given at the bottom of each design.

Appendix-2.1

A. Computer program in Microsoft Visual C++ for obtaining complete factorial experiments that are linear tend free for main effects

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
static int a[1024][10], b[1024][10], t[1024], c[1024][10], d[1024], e[1024][45],
arr[1024][45];
void main()
{
        FILE *fp;
        fp=fopen("d:\output\sush1.txt","a+");
        int i,j,h=0,l,m,n,q,k,sum=0,s,y=0,z=0;
printf("Enter the number of factors n = ");
scanf("%d",&n);
if(fp!=NULL){
m=pow(2,n);
for(i=0;i<n;i++)
        y=y+i;
for(j=0;j<n;j++)
{
  if(j==0)
  {
   for(i=0;i<m;i++)
    {
     if(i%2==0)
     a[i][j]=-1;
     else
     a[i][j]=1;
    }
```

```
    }
   else
   {
     for(i=0;i<m;)
     {
          if(h%2==0)
          {
            for(l=0;l<pow(2,j);l++)
             {
              a[i][j]=-1;
              i++;
             }
          }
          else
          {
            for(l=0;l<pow(2,j);l++)
             {
              a[i][j]=1;
              i++;
             }
          }
    h++;
    }
}
h=0;
}
if (n%2==0)
{
     for (i=0;i<m;i++)
     {
          for (k=0;k<n;k++)
```

```
           {
            b[i][k]=1;
              for(j=0;j<n;j++)
                b[i][k]= b[i][k]*a[i][j];
                b[i][k]=b[i][k]/a[i][k];
            }
}
}
else
{
    for (i=0;i<m;i++)
     {
          for (k=0;k<n-1;k++)
           {
            b[i][k]=1;
              for(j=0;j<n;j++)
                b[i][k]= b[i][k]*a[i][j];
                b[i][k]=b[i][k]/a[i][k];
            }
          {
                b[i][k]=1;
              for(j=0;j<n;j++)
                b[i][k]= b[i][k]*a[i][j];
            }
}
}
for (i=0;i<2*m;i++)
{
          q=(-(m-(i+1)));
          t[i/2]=q;
          fprintf(fp,"%3d",t[i/2]);
```

```
          fprintf(fp,"\n");
          i=i+1;
                    }
for(i=0;i<m;i++)
   {
          for(j=0;j<n;j++)
          {
          fprintf(fp,"%3d",a[i][j]);
          printf("\t");
          }
          fprintf(fp,"\n");
   }
for(i=0;i<m;i++)
   {
          for(k=0;k<n;k++)
          {
          fprintf(fp,"%3d",b[i][k]);
          fprintf(fp,"\t");
          }
          fprintf(fp,"\n");
   }
if(n==3)
{
          for(i=0;i<m;i++)
          {
                    s=b[i][0]*b[i][1];
                    printf("%3d\n",s);
          }
}
else if(n>3)
{
```

```
    for(k=0;k<n-1;k++)
    {
        for(j=1;j<n;j++)
        {
            for(i=0;i<m;i++)
            {
                arr[i][z]=b[i][k]*b[i][j];
            }
    z=z+1;
            if(z<=y)
                continue;
            else
                break;
        }
     if(z<=y)
            continue;
        else
            break;
        printf("\n");
    }
}
for(i=0;i<m;i++)
{
    for(j=0;j<y;j++)
    {
        printf("%3d",arr[i][j]);
    }
    printf("\n");
}
for(i=0;i<m;i++)
  {
```

```
            for(k=0;k<n;k++)
            {
                    c[i][k]=t[i]*b[i][k];
            fprintf(fp,"%3d",c[i][k]);
            fprintf(fp,"\t");
            }
            fprintf(fp,"\n");
   }
for(i=0;i<m;i++)
   {
            for(k=0;k<y;k++)
            {
                    e[i][k]=t[i]*arr[i][k];
            printf("%3d",e[i][k]);
            printf("\t");
            }
            printf("\n");
   }
for(k=0;k<n;k++)
   {
            for(i=0;i<m;i++)
            {
                    sum=sum+c[i][k];
            }
            fprintf(fp,"%3d",sum);
            sum=0;
            fprintf(fp,"\t");
   }
for(k=0;k<y;k++)
   {
            for(i=0;i<m;i++)
```

```
            {
                        sum=sum+e[i][k];
            }
            printf("%3d",sum);
            sum=0;
            printf("\t");
   }
}
fclose(fp);
getch();
}
```

Appendix – 2.2

Catalogue of complete factorial experiments that are linear trend-free for main effects and some of 2- and 3- factor interactions

By using the algorithm developed in section 2.3, computer program in Microsoft Visual C++ is run for obtaining complete factorial experiments that are linear-tend free for main effects. With this program, designs have been developed for *k* = 3, …, 7 factors each at two levels. These designs are linear trend-free for main effects. Two and three factor interactions that are linear trend-free (LTF) or nearly linear trend-free (NLTF) are given at the bottom of the design. For each of the design of *k* factors, *k*! different designs can be generated by permuting columns of main effects that are linear trend-free for main effects. In these *k*! different designs, the treatment combinations of two and three factor interactions that are linear/nearly linear trend-free may be different. For illustration 6 different designs Design 1.1 to Design 1.6 are given for 2^3 factorial experiment. For other design of factor *k* = 4, 5, 6 and 7 only one design is given.

2^3 Design

Design-1.1

	A	**B**	**C**
ab	1	1	-1
ac	1	-1	1
bc	-1	1	1
(1)	-1	-1	-1
c	-1	-1	1
b	-1	1	-1
a	1	-1	-1
abc	1	1	1

LTF: AB NLTF: AC

Design-1.2

	A	**B**	**C**
ac	1	-1	1
ab	1	1	-1
bc	-1	1	1
(1)	-1	-1	-1
b	-1	1	-1
c	-1	-1	1
a	1	-1	-1
abc	1	1	1

LTF: AC

Design-1.3

	A	**B**	**C**
ab	1	1	-1
bc	-1	1	1
ac	1	-1	1
(1)	-1	-1	-1
c	-1	-1	1
a	1	-1	-1
b	-1	1	-1
abc	1	1	1

LTF: AB

Design-1.4

	A	**B**	**C**
ac	1	-1	1
bc	-1	1	1
ab	1	1	-1
(1)	-1	-1	-1
b	-1	1	-1
a	1	-1	-1
c	-1	-1	1
abc	1	1	1

LTF: AC

Design-1.5

	A	**B**	**C**
bc	-1	1	1
ab	1	1	-1
ac	1	-1	1
(1)	-1	-1	-1
a	1	-1	-1
c	-1	-1	1
b	-1	1	-1
abc	1	1	1

NLTF: AB

Design-1.6

	A	**B**	**C**
bc	-1	1	1
ac	1	-1	1
ab	1	1	-1
(1)	-1	-1	-1
a	1	-1	-1
b	-1	1	-1
c	-1	-1	1
abc	1	1	1

NTLF: AC

2^4 Design

	A	B	C	D		A	B	C	D
(1)	-1	-1	-1	-1	**abc**	1	1	1	1
bcd	-1	1	1	-1	**ad**	1	-1	-1	1
acd	1	-1	1	1	**bd**	-1	1	-1	-1
ab	1	1	-1	1	**c**	-1	-1	1	-1
abd	1	1	-1	1	**cd**	-1	-1	1	-1
ac	1	-1	1	1	**b**	-1	1	-1	-1
bc	-1	1	1	-1	**a**	1	-1	-1	1
d	-1	-1	-1	-1	**abcd**	1	1	1	1

LTF: All two factor interactions NLTF: BCD

2^5 Design

	A	B	C	D	E		A	B	C	D	E
abcd	1	1	1	1	-1	**e**	-1	-1	-1	-1	1
ae	1	-1	-1	-1	1	**bce**	-1	1	1	1	-1
be	-1	1	-1	-1	1	**acd**	1	-1	1	1	-1
cde	-1	-1	1	1	-1	**abe**	1	1	-1	-1	1
ce	-1	-1	1	-1	1	**abd**	1	1	-1	1	-1
bd	-1	1	-1	1	-1	**ace**	1	-1	1	-1	1
ad	1	-1	-1	1	-1	**bce**	-1	1	1	-1	1
abce	1	1	1	-1	1	**d**	-1	-1	-1	1	-1
de	-1	-1	-1	1	1	**abc**	1	1	1	-1	-1
bc	-1	1	1	-1	-1	**ade**	1	-1	-1	1	1
ac	1	-1	1	-1	-1	**bde**	-1	1	-1	1	1
abde	1	1	-1	1	1	**c**	-1	-1	1	-1	-1
ab	1	1	-1	-1	-1	**cde**	-1	-1	1	1	1
acde	1	-1	1	1	1	**b**	-1	1	-1	-1	-1
bcde	-1	1	1	1	1	**a**	1	-1	-1	-1	-1
(1)	-1	-1	-1	-1	-1	**abcde**	1	1	1	1	1

LTF: All two and three factor interactions

2^6 Design

	A	B	C	D	E	F		A	B	C	D	E	F
(1)	-1	-1	-1	-1	-1	-1	**abcde**	1	1	1	1	1	-1
bcdef	-1	1	1	1	1	1	**af**	1	-1	-1	-1	-1	1
acdef	1	-1	1	1	1	1	**bf**	-1	1	-1	-1	-1	1
ab	1	1	-1	-1	-1	-1	**cde**	-1	-1	1	1	1	-1
abdef	1	1	-1	1	1	1	**cf**	-1	-1	1	-1	-1	1
ac	1	-1	1	-1	-1	-1	**bde**	-1	1	-1	1	1	-1
bc	-1	1	1	-1	-1	-1	**ade**	1	-1	-1	1	1	-1
def	-1	-1	-1	1	1	1	**abcf**	1	1	1	-1	-1	1
abcef	1	1	1	-1	1	1	**df**	-1	-1	-1	1	-1	1
ad	1	-1	-1	1	-1	-1	**bce**	-1	1	1	-1	1	-1
bd	-1	1	-1	1	-1	-1	**ace**	1	-1	1	-1	1	-1
cef	-1	-1	1	-1	1	1	**abdf**	1	1	-1	1	-1	1

cd	-1	-1	1	1	-1	-1	**abe**	1	1	-1	-1	1	-1
bef	-1	1	-1	-1	1	1	**acdf**	1	-1	1	1	-1	1
aef	1	-1	-1	-1	1	1	**bcdf**	-1	1	1	1	-1	1
abcd	1	1	1	1	-1	-1	**e**	-1	-1	-1	-1	1	-1
abcdf	1	1	1	1	-1	1	**ef**	-1	-1	-1	-1	1	1
ae	1	-1	-1	-1	1	-1	**bcd**	-1	1	1	1	-1	-1
be	-1	1	-1	-1	1	-1	**acd**	1	-1	1	1	-1	-1
cef	-1	-1	1	1	-1	1	**abef**	1	1	-1	-1	1	1
ce	-1	-1	1	-1	1	-1	**abd**	1	1	-1	1	-1	-1
bdf	-1	1	-1	1	-1	1	**acef**	1	-1	1	-1	1	1
adf	1	-1	-1	1	-1	1	**bcef**	-1	1	1	-1	1	1
abce	1	1	1	-1	1	-1	**d**	-1	-1	-1	1	-1	-1
de	-1	-1	-1	1	1	-1	**abc**	1	1	1	-1	-1	-1
bcf	-1	1	1	-1	-1	1	**adef**	1	-1	-1	1	1	1
acf	1	-1	1	-1	-1	1	**bdef**	-1	1	-1	1	1	1
abde	1	1	-1	1	1	-1	**c**	-1	-1	1	-1	-1	-1
abf	1	1	-1	-1	-1	1	**cdef**	-1	-1	1	1	1	1
acde	1	-1	1	1	1	-1	**b**	-1	1	-1	-1	-1	-1
bcde	-1	1	1	1	1	-1	**a**	1	-1	-1	-1	-1	-1
f	-1	-1	-1	-1	-1	1	**abcdef**	1	1	1	1	1	1

LTF: All two and three factor interactions

2^7 Design

	A	**B**	**C**	**D**	**E**	**F**	**G**		**A**	**B**	**C**	**D**	**E**	**F**	**G**
abcdef	1	1	1	1	1	1	-1	**g**	-1	-1	-1	-1	-1	-1	1
ag	1	-1	-1	-1	-1	-1	1	**bcdef**	-1	1	1	1	1	1	-1
bg	-1	1	-1	-1	-1	-1	1	**acdef**	1	-1	1	1	1	1	-1
cdef	-1	-1	1	1	1	1	-1	**abg**	1	1	-1	-1	-1	-1	1
cg	-1	-1	1	-1	-1	-1	1	**abdef**	1	1	-1	1	1	1	-1
bdef	-1	1	-1	1	1	1	-1	**acg**	1	-1	1	-1	-1	-1	1
adef	1	-1	-1	1	1	1	-1	**bcg**	-1	1	1	-1	-1	-1	1
abcg	1	1	1	-1	-1	-1	1	**def**	-1	-1	-1	1	1	1	-1
dg	-1	-1	-1	1	-1	-1	1	**abcef**	1	1	1	-1	1	1	-1
bcef	-1	1	1	-1	1	1	-1	**adg**	1	-1	-1	1	-1	-1	1
acef	1	-1	1	-1	1	1	-1	**bdg**	-1	1	-1	1	-1	-1	1
abdg	1	1	-1	1	-1	-1	1	**cef**	-1	-1	1	-1	1	1	-1
abef	1	1	-1	-1	1	1	-1	**cdg**	-1	-1	1	1	-1	-1	1
acdg	1	-1	1	1	-1	-1	1	**bef**	-1	1	-1	-1	1	1	-1
bcdg	-1	1	1	1	-1	-1	1	**aef**	1	-1	-1	-1	1	1	-1
ef	-1	-1	-1	-1	1	1	-1	**abcdg**	1	1	1	1	-1	-1	1
eg	-1	-1	-1	-1	1	-1	1	**abcdf**	1	1	1	1	-1	1	-1
bcdf	-1	1	1	1	-1	1	-1	**aeg**	1	-1	-1	-1	1	-1	1
acdf	1	-1	1	1	-1	1	-1	**beg**	-1	1	-1	-1	1	-1	1
abeg	1	1	-1	-1	1	-1	1	**cdf**	-1	-1	1	1	-1	1	-1
abdf	1	1	-1	1	-1	1	-1	**ceg**	-1	-1	1	-1	1	-1	1
aceg	1	-1	1	-1	1	-1	1	**bdf**	-1	1	-1	1	-1	1	-1
bceg	-1	1	1	-1	1	-1	1	**adf**	1	-1	-1	1	-1	1	-1
df	-1	-1	-1	1	-1	1	-1	**abceg**	1	1	1	-1	1	-1	1
abcf	1	1	1	-1	-1	1	-1	**deg**	-1	-1	-1	1	1	-1	1
adeg	1	-1	-1	1	1	-1	1	**bcf**	-1	1	1	-1	-1	1	-1
bdeg	-1	1	-1	1	1	-1	1	**acf**	1	-1	1	-1	-1	1	-1
cf	-1	-1	1	-1	-1	1	-1	**abdeg**	1	1	-1	1	1	-1	1
cdeg	-1	-1	1	1	1	-1	1	**abf**	1	1	-1	-1	-1	1	-1

bf	-1	1	-1	-1	-1	1	-1
af	1	-1	-1	-1	-1	1	-1
abcdeg	1	1	1	1	1	-1	1
fg	-1	-1	-1	-1	-1	1	1
bcde	-1	1	1	1	1	-1	-1
acde	1	-1	1	1	1	-1	-1
abfg	1	1	-1	-1	-1	1	1
abde	1	1	-1	1	1	-1	-1
acfg	1	-1	1	-1	-1	1	1
bcfg	-1	1	1	-1	-1	1	1
de	-1	-1	-1	1	1	-1	-1
abce	1	1	1	-1	1	-1	-1
adfg	1	-1	-1	1	-1	1	1
bdfg	-1	1	-1	1	-1	1	1
ce	-1	-1	1	-1	1	-1	-1
cdfg	-1	-1	1	1	-1	1	1
be	-1	1	-1	-1	1	-1	-1
ae	1	-1	-1	-1	1	-1	-1
abcdfg	1	1	1	1	-1	1	1
abcd	1	1	1	1	-1	-1	-1
aefg	1	-1	-1	-1	1	1	1
befg	-1	1	-1	-1	1	1	1
cd	-1	-1	1	1	-1	-1	-1
cefg	-1	-1	1	-1	1	1	1
bd	-1	1	-1	1	-1	-1	-1
ad	1	-1	-1	1	-1	-1	-1
abcefg	1	1	1	-1	1	1	1
defg	-1	-1	-1	1	1	1	1
bc	-1	1	1	-1	-1	-1	-1
ac	1	-1	1	-1	-1	-1	-1
abdefg	1	1	-1	1	1	1	1
ab	1	1	-1	-1	-1	-1	-1
acdefg	1	-1	1	1	1	1	1
bcdefg	-1	1	1	1	1	1	1
(1)	-1	-1	-1	-1	-1	-1	-1

acdeg	1	-1	1	1	1	-1	1
bcdeg	-1	1	1	1	1	-1	1
f	-1	-1	-1	-1	-1	1	-1
abcde	1	1	1	1	1	-1	-1
afg	1	-1	-1	-1	-1	1	1
bfg	-1	1	-1	-1	-1	1	1
cde	-1	-1	1	1	1	-1	-1
cfg	-1	-1	1	-1	-1	1	1
bde	-1	1	-1	1	1	-1	-1
ade	1	-1	-1	1	1	-1	-1
abcfg	1	1	1	-1	-1	1	1
dfg	-1	-1	-1	1	-1	1	1
bce	-1	1	1	-1	1	-1	-1
ace	1	-1	1	-1	1	-1	-1
abdfg	1	1	-1	1	-1	1	1
abe	1	1	-1	-1	1	-1	-1
acdfg	1	-1	1	1	-1	1	1
bcdfg	-1	1	1	1	-1	1	1
efg	-1	-1	-1	-1	1	-1	-1
efg	-1	-1	-1	-1	1	1	1
bcd	-1	1	1	1	-1	-1	-1
acd	1	-1	1	1	-1	-1	-1
abefg	1	1	-1	-1	1	1	1
abd	1	1	-1	1	-1	-1	-1
acefg	1	-1	1	-1	1	1	1
bcefg	-1	1	1	-1	1	1	1
d	-1	-1	-1	1	-1	-1	-1
abc	1	1	1	-1	-1	-1	-1
adefg	1	-1	-1	1	1	1	1
bdefg	-1	1	-1	1	1	1	1
c	-1	-1	1	-1	-1	-1	-1
cdefg	-1	-1	1	1	1	1	1
b	-1	1	-1	-1	-1	-1	-1
a	1	-1	-1	-1	-1	-1	-1
abcdefg	1	1	1	1	1	1	1

LTF: All two and three factor interactions

Appendix-2.3

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
static int a[1024][10], b[10], b1[10], b2[10],t[1024];
int a1[1024][10], a2[1024][10], a3[1024][10],c1[1024][10],d1[1024][10];
void main()
{
        FILE *fp;
        fp=fopen(“d:\obj2.txt”,”a+”);
        int
i,d,j=0,h=0,q,l,m,m1,n,sum,y=0,z=0,n1,j1,fr=0,k=0,n2,k1=0,n3,k2=0,k3=0,c,c5;
printf(“Enter the number of factors K = “);
scanf(“%d”,&n);
if(fp!=NULL){
m=pow(2,n);
for(i=0;i<n;i++)
        y=y+I;
for(j=0;j<n;j++)
{
  if(j==0)
  {
   for(i=0;i<m;i++)
    {
     if(i%2==0)
     a[i][j]=0;
     else
     a[i][j]=1;
    }
   }
  else
  {
    for(i=0;i<m;)
    {
        if(h%2==0)
        {
         for(l=0;l<pow(2,j);l++)
          {
           a[i][j]=0;
           i++;
          }
        }
        else
        {
         for(l=0;l<pow(2,j);l++)
```

```
            {
            a[i][j]=1;
            i++;
            }
          }
   h++;
   }
}
h=0;
}
printf("Enter 1 for half fraction");
printf("Enter 2 for quarter fraction");
printf("Enter 3 for 1/8 fraction");
scanf("%d",&fr);
if(fr==1)
{
         printf("Enter the number of factors for interaction");
         scanf("%d",&n1);
         printf("Enter interaction");
         for(i=0;i<n1;i++)
         {
                  scanf("%d",&b[i]);
         }
         for (i=0;i<2*m;i++)
         {
         q=(-(m-(i+1)));
         t[i/2]=q;
         fprintf(fp,"%3d",t[i/2]);
         fprintf(fp,"\n");
         i++;
                  }
         for(i=0;i<m;i++)
   {
                  sum=0;
   for(j1=0;j1<n1;j1++)
         {
                  sum=sum+a[i][b[j1]];
         }
                  if((sum%2)==0)
                           {
                                    for(j=0;j<n;j++)
                                             {
                                             a1[k][j]=a[i][j];

                                             }
                                             k++;
```

```
                    }
          }
for(i=0;i<n1;i++)
{
          fprintf(fp,"\n%3d",b[i]);
}
     for(k=0;k<m/2;k++)
          {
          for(j=0;j<n;j++)
               {
                    fprintf(fp,"%3d",a1[k][j]);
                    fprintf(fp,"\t");
               }
            fprintf(fp,"\n");
          }

     for(k=0;k<m/2;k++)
          {
          for(j=0;j<n;j++)
               {
                    fprintf(fp,"%3d",a1[k][j]);
                    fprintf(fp, »\t ») ;
               }
            fprintf(fp, »\n ») ;
          }
          for(k=0;k<m/2;k++)
          {

          for(j=0;j<n;j++)
               {
               if(a1[k][j]==0)
                    {
                    d=1 ;
                    fprintf(fp, »%3d »,d) ;
                    fprintf(fp, »\t ») ;
                    }
               else
               {
                    d=0;
                    fprintf(fp,"%3d",d);
                    fprintf(fp, »\t ») ;
               }
               }
            fprintf(fp, »\n ») ;
          }
     for(k=0;k<m/2;k++)
```

```
        {
        for(j=0;j<n;j++)
                {
                if(a1[k][j]==0)
                        {
                        a1[k][j]=-1;
                        c1[k][j]=a1[k][j];
                        c=c1[k][j];
                        fprintf(fp,”%3d”,c);
                        fprintf(fp,”\t”);
                        }
                else
                {
                        fprintf(fp,”%3d”,a1[k][j]);
                        fprintf(fp, »\t ») ;
                }
                }
           fprintf(fp, »\n ») ;
        }
for(k=0;k<m/2;k++)
        {
        for(j=0;j<n;j++)
                {
                if(a1[k][j]==0)
                        {
                        a1[k][j]=-1;
                        c1[k][j]=a1[k][j];
                        fprintf(fp,”%3d”,a1[k][j]);
                        fprintf(fp,”\t”);
                        }
                else
                {
                        c1[k][j]=a1[k][j];
                        fprintf(fp,”%3d”,a1[k][j]);
                        fprintf(fp, »\t ») ;
                }
                }
           fprintf(fp, »\n ») ;
        }
k3=m/2;
for(k=0;k<m/2;k++)
        {

        for(j=0;j<n;j++)
                {
                if(a1[k][j]==0)
```

```
                                {
                                a1[k][j]=-1;
                                c=-1*a1[k][j];
                                c1[k3][j]=c;
                                fprintf(fp,”%3d”,c);
                                fprintf(fp,”\t”);
                                }
                        else
                        {
                                c=-1*a1[k][j];
                                c1[k3][j]=c;
                                fprintf(fp,”%3d”,c);
                                fprintf(fp,”\t”);
                        }
                        }
                k3++;
                   fprintf(fp,”\n”);
                }
                for(k=0;k<m;k++)
        {

                for(j=0;j<n;j++)
                {
                        c1[k][j]=t[k]*c1[k][j];
                        fprintf(fp,”%3d”,c1[k][j]);
                        fprintf(fp, »\t ») ;
                }
                fprintf(fp, »\n ») ;
        }
        sum=0;
for(k=0;k<n;k++)
   {
        for(i=0;i<m;i++)
        {
                sum=sum+c1[i][k];
        }
        fprintf(fp,”%3d”,sum);
        sum=0;
        fprintf(fp,”\t”);
   }
}
if(fr==2)
{
        printf(“Enter the number of factors for first interaction”);
        scanf(“%d”,&n1);
        printf(“Enter interaction”);
```

```
        for(i=0;i<n1;i++)
        {
                scanf(“%d”,&b[i]);
        }
        printf(“Enter the number of factors for second interaction”);
        scanf(“%d”,&n2);
        printf(“Enter interaction”);
        for(i=0;i<n2;i++)
        {
                scanf(“%d”,&b1[i]);
        }
        m1=m/2;
        for (i=0;i<2*m1;i++)
        {
        q=(-(m1-(i+1)));
        t[i/2]=q;
        fprintf(fp,”%3d”,t[i/2]);
        fprintf(fp,”\n”);
        i++;
                }
        for(i=0;i<m;i++)
  {
                sum=0;
  for(j1=0;j1<n1;j1++)
        {
                sum=sum+a[i][b[j1]];
        }
                if((sum%2)==0)
                        {
                                for(j=0;j<n;j++)
                                        {
                                        a1[k][j]=a[i][j];

                                        }
                                        k++;
                        }
        }
for(i=0;i<n1;i++)
{
                fprintf(fp,”\n%3d”,b[i]);
}
        for(k=0;k<m/2;k++)
                {
                for(j=0;j<n;j++)
                        {
                        fprintf(fp,”%3d”,a1[k][j]);
```

```
                              fprintf(fp, »\t ») ;
                              }
                          fprintf(fp, »\n ») ;
                      }
             for(k=0;k<m/2;k++)
   {
                      sum=0;
   for(j1=0;j1<n2;j1++)
             {
                      sum=sum+a1[k][b1[j1]];
             }
                      if((sum%2)==0)
                              {
                                        for(j=0;j<n;j++)
                                                  {
                                                  a2[k1][j]=a1[k][j];

                                                  }
                                                  k1++;
                              }
             }
fprintf(fp,"\nb1\n");
for(i=0;i<n2;i++)
{
                      fprintf(fp,"\n%3d",b1[i]);
}
             for(k1=0;k1<m/4;k1++)
                      {
                      for(j=0;j<n;j++)
                              {
                                        fprintf(fp,"%3d",a2[k1][j]);
                                        fprintf(fp, »\t ») ;
                              }
                                 fprintf(fp, »\n ») ;
                      }
for(k1=0;k1<m/4;k1++)
                      {
                      for(j=0;j<n;j++)
                              {
                                        fprintf(fp,"%3d",a2[k1][j]);
                                        fprintf(fp, »\t ») ;
                              }
                                 fprintf(fp, »\n ») ;
                      }
for(k1=0;k1<m/4;k1++)
                      {
```

```
    for(j=0;j<n;j++)
        {
        if(a2[k1][j]==0)
        {
            d=1 ;
            fprintf(fp, »%3d »,d) ;
            fprintf(fp, »\t ») ;
        }
        else
        {
            d=0;
            fprintf(fp,”%3d”,d);
            fprintf(fp, »\t ») ;
        }
    }
         fprintf(fp, »\n ») ;
    }
for(k1=0;k1<m/4;k1++)
    {
    for(j=0;j<n;j++)
        {
        if(a2[k1][j]==0)
            {
            a2[k1][j]=-1;
            fprintf(fp,”%3d”,a2[k1][j]);
            fprintf(fp,”\t”);
            }
        else
        {
            fprintf(fp,”%3d”,a2[k1][j]);
            fprintf(fp, »\t ») ;
        }
    }
         fprintf(fp, »\n ») ;
    }
for(k1=0;k1<m/4;k1++)
    {
    for(j=0;j<n;j++)
        {
        if(a2[k1][j]==0)
            {
            a2[k1][j]=-1;
            c1[k1][j]=a2[k1][j];
            fprintf(fp,”%3d”,a2[k1][j]);
            fprintf(fp,”\t”);
            }
```

```
            else
            {
                c1[k1][j]=a2[k1][j];
                fprintf(fp,”%3d”,a2[k1][j]);
                fprintf(fp, »\t ») ;
            }
        }
             fprintf(fp, »\n ») ;
        }
        k3=m/4;
    for(k1=0;k1<m/4;k1++)
        {
        for(j=0;j<n;j++)
            {
            if(a2[k1][j]==0)
                {
                a2[k1][j]=-1;
                c=a2[k1][j]*-1;
                c1[k3][j]=c;
                fprintf(fp,”%3d”,c);
                fprintf(fp,”\t”);
                }
            else
            {
                c=a2[k1][j]*-1;
                c1[k3][j]=c;
                fprintf(fp,”%3d”,c);
                fprintf(fp,”\t”);
            }
        }k3++;
             fprintf(fp,”\n”);
        }

    for(k=0;k<m/2;k++)
    {

        for(j=0;j<n;j++)
        {
            c1[k][j]=t[k]*c1[k][j];
            fprintf(fp,”%3d”,c1[k][j]);
            fprintf(fp, »\t ») ;
        }
        fprintf(fp, »\n ») ;
    }
    sum=0;
for(k=0;k<n;k++)
```

```
   {
        for(i=0;i<m/2;i++)
        {
                sum=sum+c1[i][k];

        }
        fprintf(fp,”%3d”,sum);
        sum=0;
        fprintf(fp,”\t”);
   }
}
if(fr==3)
{
        printf(“Enter the number of factors for first interaction”);
        scanf(“%d”,&n1);
        printf(“Enter interaction”);
        for(i=0;i<n1;i++)
        {
                scanf(“%d”,&b[i]);
        }
        printf(“Enter the number of factors for second interaction”);
        scanf(“%d”,&n2);
        printf(“Enter interaction”);
        for(i=0;i<n2;i++)
        {
                scanf(“%d”,&b1[i]);
        }
        printf(“Enter the number of factors for third interaction”);
        scanf(“%d”,&n3);
        printf(“Enter interaction”);
        for(i=0;i<n3;i++)
        {
                scanf(“%d”,&b2[i]);
        }
        m1=m/4;
        for (i=0;i<2*m1;i++)
        {
        q=(-(m1-(i+1)));
        t[i/2]=q;
        fprintf(fp,”%3d”,t[i/2]);
        fprintf(fp,”\n”);
        i++;
                }
                for(i=0;i<m;i++)
   {
                sum=0;
```

```
   for(j1=0;j1<n1;j1++)
         {
                  sum=sum+a[i][b[j1]];
         }
                  if((sum%2)==0)
                           {
                                    for(j=0;j<n;j++)
                                             {
                                             a1[k][j]=a[i][j];

                                             }
                                             k++;
                           }
         }

fprintf(fp,"\nb\n");
         for(i=0;i<n1;i++)
{
                  fprintf(fp,"\n%3d",b[i]);
}
         for(k=0;k<m/2;k++)
                  {
                  for(j=0;j<n;j++)
                           {
                           fprintf(fp,"%3d",a1[k][j]);
                           fprintf(fp,"\t");
                           }
                     fprintf(fp,"\n");
                  }

         for(k=0;k<m/2;k++)
   {
                  sum=0;
   for(j1=0;j1<n2;j1++)
         {
                  sum=sum+a1[k][b1[j1]];
         }
                  if((sum%2)==0)
                           {
                                    for(j=0;j<n;j++)
                                             {
                                             a2[k1][j]=a1[k][j];

                                             }
                                             k1++;
                           }
```

```
          }

fprintf(fp,"\nb1\n");
for(i=0;i<n2;i++)
{
                    fprintf(fp,"\n%3d",b1[i]);
}
          for(k1=0;k1<m/4;k1++)
                    {
                    for(j=0;j<n;j++)
                              {
                              fprintf(fp,"%3d",a2[k1][j]);
                              fprintf(fp, »\t ») ;
                              }
                       fprintf(fp, »\n ») ;
                    }

          for(k1=0;k1<m/4;k1++)
   {
                    sum=0;
   for(j1=0;j1<n3;j1++)
          {
                    sum=sum+a2[k1][b2[j1]];
          }
                    if((sum%2)==0)
                              {
                                        for(j=0;j<n;j++)
                                                  {
                                                  a3[k2][j]=a2[k1][j];

                                                  }
                                                  k2++;
                              }
          }

fprintf(fp,"\nb2\n");
for(i=0;i<n3;i++)
{
                    fprintf(fp,"\n%3d",b2[i]);
}

          for(k2=0;k2<m/8;k2++)
                    {
                    for(j=0;j<n;j++)
                              {
                                        fprintf(fp,"%3d",a3[k2][j]);
```

```
                                fprintf(fp,”\t”);
                        }

                   fprintf(fp,”\n”);
                }
        for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                                fprintf(fp,”%3d”,a3[k2][j]);
                                fprintf(fp, »\t ») ;
                        }
                   fprintf(fp, »\n ») ;
                }
for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                        if(a3[k2][j]==0)
                        {
                                d=1 ;
                                fprintf(fp, »%3d »,d) ;
                                fprintf(fp, »\t ») ;
                        }
                        else
                        {
                                d=0;
                                fprintf(fp,”%3d”,d);
                                fprintf(fp, »\t ») ;
                        }
                }
                   fprintf(fp, »\n ») ;
                }
        for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                        if(a3[k2][j]==0)
                                {
                                a3[k2][j]=-1;
                                fprintf(fp,”%3d”,a3[k2][j]);
                                fprintf(fp,”\t”);
                                }
                        else
                        {
                                fprintf(fp,”%3d”,a3[k2][j]);
```

```
                                fprintf(fp, »\t ») ;
                        }
                        }

                   fprintf(fp, »\n ») ;
                }
for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                        if(a3[k2][j]==0)
                                {
                                a3[k2][j]=-1;
                                c1[k2][j]=a3[k2][j];
                                fprintf(fp,”%3d”,a3[k2][j]);
                                fprintf(fp,”\t”);
                                }
                        else
                        {
                                c1[k2][j]=a3[k2][j];
                                fprintf(fp,”%3d”,a3[k2][j]);
                                fprintf(fp, »\t ») ;
                        }
                        }

                   fprintf(fp, »\n ») ;
                }
        k3=m/8;
        for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                        if(a3[k2][j]==0)
                                {
                                a3[k2][j]=-1;
                                c=a3[k][j]*-1;
                                c1[k3][j]=c;
                                fprintf(fp,”%3d”,c);
                                fprintf(fp,”\t”);
                                }
                        else
                        {
                                c=a3[k2][j]*-1;
                                c1[k3][j]=c;
                                fprintf(fp,”%3d”,c);
                                fprintf(fp,”\t”);
```

```
                        }
                        }k3++;

                   fprintf(fp,”\n”);
                }

        for(k=0;k<m/4;k++)
        {
                for(j=0;j<n;j++)
                {
                        c1[k][j]=t[k]*c1[k][j];
                        fprintf(fp,”%3d”,c1[k][j]);
                        fprintf(fp, »\t ») ;
                }
                fprintf(fp, »\n ») ;
        }

sum=0;
for(k=0;k<n;k++)
   {
        for(i=0;i<m/4;i++)
        {
                sum=sum+c1[i][k];
        }
        fprintf(fp,”%3d”,sum);
        sum=0;
        fprintf(fp,”\t”);
   }
}
fprintf(fp,”\nfr=%d “,fr);
}
fclose(fp);
getch();
                                              }
```

Appendix-2.4

Catalogue of confounded factorial experiments that are linear trend-free for main effects and some of 2- and 3- factor interactions

2^4 Design, confounding ABCD

Block-1					Block-2				
A	**B**	**C**	**D**		**A**	**B**	**C**	**D**	
-1	-1	-1	-1	(1)	-1	1	1	1	bcd
1	1	-1	-1	ab	1	-1	1	1	acd
1	-1	1	-1	ac	1	1	-1	1	abd
-1	1	1	-1	bc	-1	-1	-1	1	d
1	-1	-1	1	ad	1	1	1	-1	abc
-1	1	-1	1	bd	-1	-1	1	-1	c
-1	-1	1	1	cd	-1	1	-1	-1	b
1	1	1	1	abcd	1	-1	-1	-1	a

LTF: All two factor interaction and BCD; NLTF: ACD

2^5 Design, confounding ABCDE

Block-1					Block-2				
A	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
-1	-1	-1	-1	-1	1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	1	-1	-1	-1
1	-1	1	-1	-1	-1	-1	1	-1	-1
-1	1	1	-1	-1	1	1	1	-1	-1
1	-1	-1	1	-1	-1	-1	-1	1	-1
-1	1	-1	1	-1	1	1	-1	1	-1
-1	-1	1	1	-1	1	-1	1	1	-1
1	1	1	1	-1	-1	1	1	1	-1
1	-1	-1	-1	1	-1	-1	-1	-1	1
-1	1	-1	-1	1	1	1	-1	-1	1
-1	-1	1	-1	1	1	-1	1	-1	1
1	1	1	-1	1	-1	1	1	-1	1
-1	-1	-1	1	1	1	-1	-1	1	1
1	1	-1	1	1	-1	1	-1	1	1
1	-1	1	1	1	-1	-1	1	1	1
-1	1	1	1	1	1	1	1	1	1

LTF: AB, AC, AD, AE, BC, BD, CD and all three factor interactions; NLTF: BE

2^6 Design, confounding ABC, DEF and ABCDEF

	Block-1							Block-2					
	A	B	C	D	E	F		A	B	C	D	E	F
(1)	-1	-1	-1	-1	-1	-1	abcef	1	1	1	-1	1	1
ab	1	1	-1	-1	-1	-1	cef	-1	-1	1	-1	1	1
ac	1	-1	1	-1	-1	-1	bef	-1	1	-1	-1	1	1

bc	-1	1	1	-1	-1	-1
de	-1	-1	-1	1	1	-1
abde	1	1	-1	1	1	-1
acde	1	-1	1	1	1	-1
bcde	-1	1	1	1	1	-1
df	-1	-1	-1	1	-1	1
abdf	1	1	-1	1	-1	1
acdf	1	-1	1	1	-1	1
bcdf	-1	1	1	1	-1	1
ef	-1	-1	-1	-1	1	1
abef	1	1	-1	-1	1	1
acef	1	-1	1	-1	1	1
bcef	-1	1	1	-1	1	1

aef	1	-1	-1	-1	1	1
abcdf	1	1	1	1	-1	1
cdf	-1	-1	1	1	-1	1
bdf	-1	1	-1	1	-1	1
adf	1	-1	-1	1	-1	1
abcde	1	1	1	1	1	-1
cde	-1	-1	1	1	1	-1
bde	-1	1	-1	1	1	-1
ade	1	-1	-1	1	1	-1
abc	1	1	1	-1	-1	-1
c	-1	-1	1	-1	-1	-1
b	-1	1	-1	-1	-1	-1
a	1	-1	-1	-1	-1	-1

Block-3

	A	B	C	D	E	F
bcdef	-1	1	1	1	1	1
acdef	1	-1	1	1	1	1
abdef	1	1	-1	1	1	1
def	-1	-1	-1	1	1	1
bcf	-1	1	1	-1	-1	1
acf	1	-1	1	-1	-1	1
abf	1	1	-1	-1	-1	1
f	-1	-1	-1	-1	-1	1
bce	-1	1	1	-1	1	-1
ace	1	-1	1	-1	1	-1
abe	1	1	-1	-1	1	-1
e	-1	-1	-1	-1	1	-1
bcd	-1	1	1	1	-1	-1
acd	1	-1	1	1	-1	-1
abd	1	1	-1	1	-1	-1
d	-1	-1	-1	1	-1	-1

Block-4

	A	B	C	D	E	F
ad	1	-1	-1	1	-1	-1
bd	-1	1	-1	1	-1	-1
cd	-1	-1	1	1	-1	-1
abcd	1	1	1	1	-1	-1
ae	1	-1	-1	-1	1	-1
be	-1	1	-1	-1	1	-1
ce	-1	-1	1	-1	1	-1
abce	1	1	1	-1	1	-1
af	1	-1	-1	-1	-1	1
bf	-1	1	-1	-1	-1	1
cf	-1	-1	1	-1	-1	1
abcf	1	1	1	-1	-1	1
adef	1	-1	-1	1	1	1
bdef	-1	1	-1	1	1	1
cdef	-1	-1	1	1	1	1
abcdef	1	1	1	1	1	1

LTF: All two and three factor interactions

2^7 Design, confounding ABCDE, CDEFG and ABEFG

Block-1

	A	B	C	D	E	F	G
abcdef	1	1	1	1	1	1	-1
cdef	-1	-1	1	1	1	1	-1
abef	1	1	-1	-1	1	1	-1
ef	-1	-1	-1	-1	1	1	-1
abdf	1	1	-1	1	-1	1	-1
df	-1	-1	-1	1	-1	1	-1
abcf	1	1	1	-1	-1	1	-1
cf	-1	-1	1	-1	-1	1	-1
acfg	1	-1	1	-1	-1	1	1
bcfg	-1	1	1	-1	-1	1	1
adfg	1	-1	-1	1	-1	1	1
bdfg	-1	1	-1	1	-1	1	1
aefg	1	-1	-1	-1	1	1	1
befg	-1	1	-1	-1	1	1	1
acdefg	1	-1	1	1	1	1	1

Block-2

	A	B	C	D	E	F	G
bdef	-1	1	-1	1	1	1	-1
adef	1	-1	-1	1	1	1	-1
bcef	-1	1	1	-1	1	1	-1
acef	1	-1	1	-1	1	1	-1
bcdf	-1	1	1	1	-1	1	-1
acdf	1	-1	1	1	-1	1	-1
bf	-1	1	-1	-1	-1	1	-1
af	1	-1	-1	-1	-1	1	-1
fg	-1	-1	-1	-1	-1	1	1
abfg	1	1	-1	-1	-1	1	1
cdfg	-1	-1	1	1	-1	1	1
abcdfg	1	1	1	1	-1	1	1
cefg	-1	-1	1	-1	1	1	1
abcefg	1	1	1	-1	1	1	1
defg	-1	-1	-1	1	1	1	1

bcdefg	-1	1	1	1	1	1	1
acg	1	-1	1	-1	-1	-1	1
bcg	-1	1	1	-1	-1	-1	1
adg	1	-1	-1	1	-1	-1	1
bdg	-1	1	-1	1	-1	-1	1
aeg	1	-1	-1	-1	1	-1	1
beg	-1	1	-1	-1	1	-1	1
acdeg	1	-1	1	1	1	-1	1
bcdeg	-1	1	1	1	1	-1	1
abcde	1	1	1	1	1	-1	-1
cde	-1	-1	1	1	1	-1	-1
abe	1	1	-1	-1	1	-1	-1
e	-1	-1	-1	-1	1	-1	-1
abd	1	1	-1	1	-1	-1	-1
d	-1	-1	-1	1	-1	-1	-1
abc	1	1	1	-1	-1	-1	-1
c	-1	-1	1	-1	-1	-1	-1

abdefg	1	1	-1	1	1	1	1
g	-1	-1	-1	-1	-1	-1	1
abg	1	1	-1	-1	-1	-1	1
cdg	-1	-1	1	1	-1	-1	1
abcdg	1	1	1	1	-1	-1	1
ceg	-1	-1	1	-1	1	-1	1
abceg	1	1	1	-1	1	-1	1
deg	-1	-1	-1	1	1	-1	1
abdeg	1	1	-1	1	1	-1	1
bde	-1	1	-1	1	1	-1	-1
ade	1	-1	-1	1	1	-1	-1
bce	-1	1	1	-1	1	-1	-1
ace	1	-1	1	-1	1	-1	-1
bcd	-1	1	1	1	-1	-1	-1
acd	1	-1	1	1	-1	-1	-1
b	-1	1	-1	-1	-1	-1	-1
a	1	-1	-1	-1	-1	-1	-1

Block-3

	A	**B**	**C**	**D**	**E**	**F**	**G**
ag	1	-1	-1	-1	-1	-1	1
bg	-1	1	-1	-1	-1	-1	1
acdg	1	-1	1	1	-1	-1	1
bcdg	-1	1	1	1	-1	-1	1
aceg	1	-1	1	-1	1	-1	1
bceg	-1	1	1	-1	1	-1	1
adeg	1	-1	-1	1	1	-1	1
bdeg	-1	1	-1	1	1	-1	1
abde	1	1	-1	1	1	-1	-1
de	-1	-1	-1	1	1	-1	-1
abce	1	1	1	-1	1	-1	-1
ce	-1	-1	1	-1	1	-1	-1
abcd	1	1	1	1	-1	-1	-1
cd	-1	-1	1	1	-1	-1	-1
ab	1	1	-1	-1	-1	-1	-1
(1)	-1	-1	-1	-1	-1	-1	-1
abdef	1	1	-1	1	1	1	-1
def	-1	-1	-1	1	1	1	-1
abcef	1	1	1	-1	1	1	-1
cef	-1	-1	1	-1	1	1	-1
abcdf	1	1	1	1	-1	1	-1
cdf	-1	-1	1	1	-1	1	-1
abf	1	1	-1	-1	-1	1	-1
f	-1	-1	-1	-1	-1	1	-1
afg	1	-1	-1	-1	-1	1	1
bfg	-1	1	-1	-1	-1	1	1
acdfg	1	-1	1	1	-1	1	1
bcdfg	-1	1	1	1	-1	1	1
acefg	1	-1	1	-1	1	1	1
bcefg	-1	1	1	-1	1	1	1
adefg	1	-1	-1	1	1	1	1
bdefg	-1	1	-1	1	1	1	1

Block-4

	A	**B**	**C**	**D**	**E**	**F**	**G**
cg	-1	-1	1	-1	-1	-1	1
abcg	1	1	1	-1	-1	-1	1
dg	-1	-1	-1	1	-1	-1	1
abdg	1	1	-1	1	-1	-1	1
eg	-1	-1	-1	-1	1	-1	1
abeg	1	1	-1	-1	1	-1	1
cdeg	-1	-1	1	1	1	-1	1
abcdeg	1	1	1	1	1	-1	1
bcde	-1	1	1	1	1	-1	-1
acde	1	-1	1	1	1	-1	-1
be	-1	1	-1	-1	1	-1	-1
ae	1	-1	-1	-1	1	-1	-1
bd	-1	1	-1	1	-1	-1	-1
ad	1	-1	-1	1	-1	-1	-1
bc	-1	1	1	-1	-1	-1	-1
ac	1	-1	1	-1	-1	-1	-1
bcdef	-1	1	1	1	1	1	-1
acdef	1	-1	1	1	1	1	-1
bef	-1	1	-1	-1	1	1	-1
aef	1	-1	-1	-1	1	1	-1
bdf	-1	1	-1	1	-1	1	-1
adf	1	-1	-1	1	-1	1	-1
bcf	-1	1	1	-1	-1	1	-1
acf	1	-1	1	-1	-1	1	-1
cfg	-1	-1	1	-1	-1	1	1
abcfg	1	1	1	-1	-1	1	1
dfg	-1	-1	-1	1	-1	1	1
abdfg	1	1	-1	1	-1	1	1
efg	-1	-1	-1	-1	1	1	1
abefg	1	1	-1	-1	1	1	1
cdefg	-1	-1	1	1	1	1	1
abcdefg	1	1	1	1	1	1	1

LTF: All two and three factor interactions except CD

Chapter III

DEVELOPMENT OF LINEAR TREND-FREE DESIGNS FOR TWO LEVEL FRACTIONAL FACTORIAL EXPERIMENTS

3.1 Introduction

Factorial experiments are quite useful for conducting the field experiments because these experiments give the comparison of not only main effects but all types of interactions. The comparisons required in this type of experiments are not the pair-wise comparison as in varietal trials but a special type of comparison called main effects and interactions.

In factorial experiments as the number of factors and/or levels increase the number of treatment combinations increase very rapidly e.g. in 2^k factorial experiment, when $k = 3$ total number of treatment combination $n = 8$ and when $k = 7$ total number of treatment combination becomes $n = 128$. In such situations, the resources of the experimenter may not allow to have even one replication. Finney (1945) proposed a method in which only a fraction of treatment combinations will be experimented with. Such experiments are called *fractional factorial experiments*. In fractional factorial experiments, though the size of experiment is reduced, information on certain higher order interactions is sacrificed. The crucial part of the fractionally replicated design is the suitable choice of the defining or identity relationship. The non-estimable effects or interactions for the selected fraction of the treatment combinations, when equated with defining contrasts I, are called the identity relation. After selecting a fraction of treatment combinations one can easily note that any contrast of the selected treatment combinations represents more than one effect or interaction, and all effects or interactions, represented by the same treatment combinations are called aliases. In aliases, assuming that higher order interactions are negligible when compared with one of them, in which he is interested (lower order); the experimenter can estimate it by the corresponding contrast of the selected treatment combinations.

In fractional factorial experiment, lower order interactions are more likely to be important than higher order interactions. There has been much progress on the construction of two level factorial designs. Recently more attention has been paid to regular two level-designs that are characterized by their specification in terms of defining contrasts.

Similar to complete factorial experiments, the experimental units in fractional factorial designs may exhibit trend effect over space or time. The contrasts of interests, say main effects, two factor interactions, *etc*. in fractional factorial designs may be affected by trend effect. In such situations, the interest of experimenter is to eliminate the trend effect. It is known that most of the trend is reduced, if we are able to eliminate the linear trend. Thus in the presence of linear trend in experiments, the run order of treatment combinations are made such that contrasts of interest i.e. main effects, two factor interactions, etc. are orthogonal to linear trend.

In chapter II, the algorithm for complete and confounded factorial experiment that is at least linear trend-free for main effects have been developed. These designs were further searched for two and three factor interactions that are linear trend-free. The catalogues of complete and confounded factorial experiments that are linear trend-free for at least main effects have also been given.

The concept of complement foldover is given in Section 3.2. The algorithm has been developed to obtain the fractional factorial experiments that are linear trend-free for at least main effects in Section 3.3. This algorithm will also make a search for identification of two factor interactions that are estimable free from linear trend effects. This was followed by working of algorithm, computer program and catalogue of designs obtained.

In fractional factorial experiments, we consider the problem of constructing a $\frac{1}{2^p}$ fraction of a 2^k factorial experiment. Such an experiment is denoted by 2^{k-p}. Out of the total of (2^k-1) effects and interactions in the full factorial experiment, (2^p-1) are inseparable from the mean and the remaining 2^{k-p} are mutually inseparable in the set of 2^p, there being (2^{k-p-1}-1) such sets.

A necessary and sufficient condition for a linear trend-free factorial design is that there exists a systematic run order for that factorial design satisfying equation $\mathbf{X}'\mathbf{T} = \mathbf{0}$ where columns of matrix **X** are coefficients of contrasts of interest and **T** is vector of coefficients of polynomial of first order. For example **A**, **B**, **C**, … are representing the coefficients of contrasts of main effects and the fractional factorial experiment is 2^{k-p}. The experimenter is interested in estimating these main effect contrasts that are free from linear trend. It means that the main effects are orthogonal to linear trend. In such situation, the matrix is **X** = [**A B C** …] and the trend vector **T** are such that $\mathbf{X}'\mathbf{T} = \mathbf{0}$.

The technique by which useful information can be obtained with a reasonable degree of precision from less than a full replicate of a factorial experiment is known as fractional replication. Here we will use the concept of *complement foldover* pair in factorial experiments. In a factorial experiment two treatment combinations are said to be complement foldover pair, if one is obtained as the complement of the other.

One of the methods is the Generalized Foldover Scheme (GFS) introduced by Coster and Cheng (1988). This scheme uses the concept of foldover. Another important feature discussed by Coster and Cheng (1988) is of the cost of conducting a factorial experiment. In this case, the cost of conducting an experiment refers to the cost incurred by changing the levels of factors in the experiment. For example, consider a factorial experiment in which changing the levels of some factors is technically difficult and/or expensive.

In such situation, the implication is that the more frequent the level changes of these factors, the more difficult and/or expensive it is to conduct of the factorial experiment. That is, in order to reduce the cost of conducting an experiment, one may find a systematic run order such that the number of level changes is minimized. In section 3.2, we will review the concept of complement foldover, which is a basic concept of the GFS.

3.2 Concept of complement foldover

Consider a two-level complete factorial design ***D*** with four factors, i.e., ***D*** is 2^4. This design consists of four factors, say A, B, C and D each at two levels and having 16 treatment combinations or runs. In factorial experiment, two design points such as *ab* and *cd* are complementary in the sense that one is obtained by changing the levels of all the factors, in other words it is said to be complement foldover pair. A foldover for *n* points is a design that consists of *n*/2 foldover pairs. We can foldover a design with *n* points by running other *n* points that is their complements.

The concept of complement foldover is used to obtain the fractional factorial experiments that are linear trend-free for main effects and some of the treatment combinations. First we give two Lemmas.

Lemma 3.1: If we have a factorial experiment in *n* runs (say design ***D***); take its complement foldover (say design $\boldsymbol{D}'$), then combining the two, we have design (***D****) in 2*n* runs, that is linear trend-free for main effects.

Proof: for the proof of above lemma, it is sufficient to show that sum of the positions high level and low level are equal for all the factors under consideration.

Let $n_i(0)$ and $n_i(1)$ be the number of the i^{th} runs in ***D***, in which A (factor) is at its low level and high level, respectively.

Denote the sum $\sum n_i(0)$, of the run numbers in ***D*** with A at its low level by $s_1(0)$ and the corresponding sum at its high level as $s_1(1)$ i.e. $s_1(0) = \sum n_i(0)$ and $s_1(1) = \sum n_i(1)$ in ***D***. In other words, $s_1(0)$ represents the sum of positions of low level, and $s_1(1)$ represents the sum of positions of high level, in design ***D***.

Similarly $s_1'(0)$ and $s_1'(1)$ are represented in $\boldsymbol{D}'$ and $s_1^*(0)$ and $s_1^*(1)$ are represented in ***D****. Since $\boldsymbol{D}'$ is complement foldover of ***D***, so

$$s_1'(0) = \sum n + n_i(1)$$

Then

$$s_1^*(0) = s_1(0) + s_1(1) + n^2/2 = s_1^*(1)$$

Thus the sum of positions of high and low level is same in design $\boldsymbol{D}^*$. Hence the Lemma is proved.

Lemma 3.2: If we make a design (***D****) of 2*n* runs by combining a factorial experiment (***D***) of *n* runs and ($\boldsymbol{D}'$) of *n* runs i.e. complement foldover of ***D***, then the main effects which are linear trend-free in ***D*** are quadratic trend-free in ***D****.

We illustrate Lemma 3.1 with a suitable example. Suppose we begin with half replicate of 2^3 factorial experiment defined by defining contrast I = -ABC as

D (1) ab ac bc

Its complement foldover design $\boldsymbol{D}'$ is

abc c b a

Combining the two, we have ***D**** as

(1) ab ac bc abc c b a

Now if we apply the condition of linear trend-free for main effects $\mathbf{X}'\mathbf{T} = \mathbf{0}$ for all the coefficient of contrasts of main effects, it holds true.

$$\begin{array}{cccccccccc}
 & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{T} & & & \mathbf{A} & \mathbf{B} & \mathbf{C} \\
 & -1 & -1 & -1 & -7 & & & 7 & 7 & 7 \\
 & 1 & 1 & -1 & -5 & & & -5 & -5 & 5 \\
 & 1 & -1 & 1 & -3 & & & -3 & 3 & -3 \\
\mathbf{X} = & -1 & 1 & 1 & -1 & & \mathbf{X}'\mathbf{T} = & 1 & -1 & -1 \\
 & 1 & 1 & 1 & 1 & & & 1 & 1 & 1 \\
 & -1 & -1 & 1 & 3 & & & -3 & -3 & 3 \\
 & -1 & 1 & -1 & 5 & & & -5 & 5 & -5 \\
 & 1 & -1 & -1 & 7 & & & 7 & -7 & -7 \\
 & & & & & & \mathbf{Sum} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{array}$$

We denote the first sequence of n points by ***D*** and last sequence of n points that is complementary foldover of ***D***, as $\boldsymbol{D'}$ and ***D**** be sequence of 2n points. The sequence ***D*** and $\boldsymbol{D'}$ are such that each factor appears at its high level and low levels exactly n/2 times in ***D*** and $\boldsymbol{D'}$. Then we have the following rules.

Rule1 3.1: If the factor A is folded over as we pass from ***D*** to $\boldsymbol{D'}$, we obtain design (*D**). In design *D**, the level of factor A in the $(n+i)^{\text{th}}$ position is opposite of its level in the i^{th} position. The sum of the positions of high level of A is equal to the sum of the positions of low level of A. Thus the main effect A is orthogonal to linear trend.

Rule 3.2: If A is linear trend-free in **D** then A is quadratic trend free in ***D****.

Consider a 2^{k-p} fractional factorial design, which has k factors, denoted by $1,2,\ldots,k$ and 2^{k-p} runs uniquely determined by p independent defining words (contrasts). A word consists of letters that are names of factors denoted by $1,2,\ldots,k$ or $a,b,\ldots$ and so on. The group formed by p defining words can be represented by

$\mathrm{I} = \mathrm{w}_1 = \mathrm{w}_2 = \cdots = \mathrm{w}_{\text{p-1}}$

or the treatment defining contrasts sub group $\mathrm{G} = \{\mathrm{I}, \mathrm{w}_1, \cdots, \mathrm{w}_{\text{p-1}}\}$.

The number of letters in a word is called word length.

For obtaining the design that is linear trend-free for main effects by the technique of complement foldover, we have two rules other than the rule 3.1 and rule 3.2. If we start with a defining word and obtain a half replicate design say ***D***. Its complement foldover is $\boldsymbol{D'}$. Then combining the two, we have design ***D****. Then the two rules are:

Rule 3.3: If the letters in the defining word for design ***D*** are even, then design $\boldsymbol{D'}$ is mirror image of ***D*** and so we have only as much information in ***D**** as contained in ***D*** (half replicate).

Rule 3.4: If the letters in the defining word for design ***D*** are odd, then the design $\boldsymbol{D'}$ will contain the different treatment combination as contained in ***D***. Then we have the design ***D****** that will have more information than ***D*** and will be complete factorial experiment.

For clarity, we illustrate these two rules by taking the following two different factorial experiments.

1) When the total number of factors is odd i.e. 2^3 factorial experiment

Here we consider the two situations as:

A. When letters in the defining word for design *D* are even i.e. I = AB

Half replicate of design ***D*** is

1 ab c abc

Its complement foldover design $\boldsymbol{D'}$ is

abc c ab 1

Combining the two, we have ***D**** as

1 ab c abc abc c ab 1 (3.1)

B. When letters in the defining word for design D are odd i.e. I = -ABC

Half replicate design ***D*** is

1 ab ac bc

Its complement foldover design $\boldsymbol{D'}$ is

abc c b A

Combining the two, we have ***D**** as

1 ab ac bc abc c b a (3.2)

It can be easily checked that designs ***D**** in (3.1) and (3.2) are linear tend-free for main effects (Lemma 3.1) but design in (3.1) has the information same as in half replicate design i.e. that of design ***D*** while the design in 3.2 contains the information of complete factorial experiment i.e. double of the information contained in design ***D***.

2) When the total number of factors is even i.e. 2^4 factorial experiment

Here also we consider the two different situations as:

A. When letters in the defining word for design *D* are even i.e. $I = ABCD$

Half replicate of design ***D*** is

1 ab ac bc ad bd cd Abcd

Its complement foldover design ***D′*** is

abcd cd bd ad bc ac ab 1

Combining the two, we have ***D**** as

1 ab ac bc ad bd cd abcd abcd cd bd ad bc ac ab 1 (3.3)

B. When letters in the defining word for design *D* are odd i.e. $I = -ABC$

Then half replicate design ***D*** is

1 ab ac bc d abd acd bcd

Its complement foldover design ***D′*** is

abcd cd bd ad abc C b a

Now combining the two, we have ***D**** as

1 ab ac bc d abd acd bcd abcd cd bd ad Abc c b a (3.4)

The designs ***D**** in (3.3) and (3.4) are linear trend-free for main effects (Lemma 3.1) but the design in (3.3) has the information same as in half replicate design while the design in (3.4) contains the information of complete factorial experiment.

Thus for generation of a fractional factorial experiment of desired fraction that is linear trend-free for main effects, we use the following principles:

i) For obtaining complete factorial experiment that is linear trend-free for main effects, one must start with $\frac{1}{2}$ fraction and the defining contrast must contain odd number of factors.

ii) For obtaining $\frac{1}{2}$ fraction of factorial experiment that is linear trend-free for main effects, one must start with $\frac{1}{4}$ fraction and two independent defining contrasts must contain odd number of factors.

iii) For obtaining $\frac{1}{4}$ fraction of factorial experiment that is linear trend-free for main effects, one must start with $\frac{1}{8}$ fraction and three independent defining contrasts must contain odd number of factors.

iv) The design $\boldsymbol{D}$ and $\boldsymbol{D'}$ must contain each factor at its high level and low level equal times, so that main effects are estimable.

Using the Rule 3.1 to Rule 3.4 and simultaneously above principles, the algorithm is developed to obtain desired fractional factorial experiment that is linear trend-free for at least main effects.

3.3 Algorithm for trend-free fractional factorial experiments

I. Algorithm to generate fractional factorial experiments

I.1 Use steps I.1 to I.8 of section 2.3 and generate full factorial design in standard order. The factorial experiment generated is 2^k and the levels of the design are -1 and +1.

I.2 For obtaining the fraction of 2^k factorial experiment, choose the treatment combination/ interaction to form identity group of contrast.

I.3 For solving the equations, change all -1 entries into 0.

I.4 Solve the following equations for the chosen defining contrast

$$\sum x_j = 0$$
$$\sum x_j = 1; (j = 1, 2, \ldots, k)$$

where j takes the values of chosen contrast for $(j = 1, 2, \ldots, k)$, the treatment combination to be taken in defining contrast.

I.5 Out of the two equations, retain the treatment combinations of any one, say of $\sum x_j = 0$, for having the half fraction.

I.6 Similarly other equations are solved if more than one treatment combinations are to be taken in defining contrasts by following principle of defining contrast.

I.7 If p independent treatment combinations are taken in defining contrasts, then this algorithm generates a $\frac{1}{2^p}$ replicate of size 2^{k-p}.

I.8 The fraction is taken to the order so that main effects are not alias of two factor interactions, may be alias of higher order. This ensures the estimation of main effect and two factor interactions precisely.

I.9 For half fraction of a factorial experiment, we need to obtain one quarter replicate.

Using steps I.1 to I.7 in Section 3.3, a fractional factorial of desired fraction is generated. For a fraction of the factorial experiment, say 2^{k-p}, p independent defining words (contrasts) are taken such that the treatment combination in the fraction must contain each factor at its high level and low level equal times, so that main effects are estimable.

For obtaining the fractional factorial design that is linear trend-free for main effects, we follow the Rule 3.1 to Rule 3.4 and using the above principles. Here we describe the algorithm to half replicate that is linear trend-free for main effects.

II. Algorithm for obtaining fractional factorial design that is linear trend-free for main effects.

In this section, algorithm is given to obtain a fractional factorial experiment that is linear trend-free for main effects and to identify the treatment combinations of two factor interactions that are linear trend-free/ nearly linear trend-free. Here we are not considering the three factor interactions as these may be alias of main and two factor interactions. The steps of the algorithm are given below:

II.1 Decide the defining word of even letters that is to be used for defining contrast.

II.2 Take two independent defining words of odd order length such that their interaction is the desired defining contrast as in II.1.

II.3 Taking these two odd letters defining word, generate one-fourth replication by using steps I.1 to I.7 of above algorithm.

II.4 Change all 0's by -1, say this design ***D***.

II.5 Generate the design $\boldsymbol{D'}$ by taking the complement foldover of ***D*** by replacing all 1 by -1 and -1 by 1.

II.6 Combine the design ***D*** and $\boldsymbol{D'}$. This is desired design ***D****.

II.7 The design ***D**** retains the treatment combinations of defining words of odd order as discussed in rule 3.4. Thus the ***D**** is the desired design of half fraction in which the defining word is same as in II.1.

II.7 Identify the two factor interactions that are linear trend-free/ linear trend-free using the steps III.1 to III.5 of section 2.3.

Using this algorithm, we give the working of algorithm by taking a fractional factorial experiment ½ (2^5) in Section 3.4.

3.4 Working of Algorithm

Suppose the experimenter wants a fractional factorial experiment $\frac{1}{2}(2^5)$ in which the defining contrast is I = ABDE and all the main effects A, B, C, D and E are linear trend-free. For this we identify two odd letter defining contrasts such that their generalized interaction is ABDE. These two defining contrasts are ABC and CDE. For obtaining the desired design, the working of algorithm is as follows:

First develop the factorial experiment 2^5 with factors A, B, C, D and E. This is generated using steps I.1 to I.8 of algorithm given in section 2.3. This is given in Design 3.1.

Design 3.1

	A	**B**	**C**	**D**	**E**		**A**	**B**	**C**	**D**	**E**
(1)	-1	-1	-1	-1	-1	**e**	-1	-1	-1	-1	1
a	1	-1	-1	-1	-1	**ae**	1	-1	-1	-1	1
b	-1	1	-1	-1	-1	**be**	-1	1	-1	-1	1
ab	1	1	-1	-1	-1	**abe**	1	1	-1	-1	1

c	-1	-1	1	-1	-1	**ce**	-1	-1	1	-1	1
ac	1	-1	1	-1	-1	**ace**	1	-1	1	-1	1
bc	-1	1	1	-1	-1	**bce**	-1	1	1	-1	1
abc	1	1	1	-1	-1	**abce**	1	1	1	-1	1
d	-1	-1	-1	1	-1	**de**	-1	-1	-1	1	1
ad	1	-1	-1	1	-1	**ade**	1	-1	-1	1	1
bd	-1	1	-1	1	-1	**bde**	-1	1	-1	1	1
abd	1	1	-1	1	-1	**abde**	1	1	-1	1	1
cd	-1	-1	1	1	-1	**cde**	-1	-1	1	1	1
acd	1	-1	1	1	-1	**acde**	1	-1	1	1	1
bcd	-1	1	1	1	-1	**bcde**	-1	1	1	1	1
abcd	1	1	1	1	-1	**abcde**	1	1	1	1	1

Change all -1 entries into 0 as per step I.3 of this section.

Design 3.2

	A	**B**	**C**	**D**	**E**		**A**	**B**	**C**	**D**	**E**
(1)	0	0	0	0	0	**e**	0	0	0	0	1
a	1	0	0	0	0	**ae**	1	0	0	0	1
b	0	1	0	0	0	**be**	0	1	0	0	1
ab	1	1	0	0	0	**abe**	1	1	0	0	1
c	0	0	1	0	0	**ce**	0	0	1	0	1
ac	1	0	1	0	0	**ace**	1	0	1	0	1
bc	0	1	1	0	0	**bce**	0	1	1	0	1
abc	1	1	1	0	0	**abce**	1	1	1	0	1
d	0	0	0	1	0	**de**	0	0	0	1	1
ad	1	0	0	1	0	**ade**	1	0	0	1	1
bd	0	1	0	1	0	**bde**	0	1	0	1	1
abd	1	1	0	1	0	**abde**	1	1	0	1	1
cd	0	0	1	1	0	**cde**	0	0	1	1	1
acd	1	0	1	1	0	**acde**	1	0	1	1	1
bcd	0	1	1	1	0	**bcde**	0	1	1	1	1
abcd	1	1	1	1	0	**abcde**	1	1	1	1	1

For half fraction, we take the Defining contrasts as ABC *i.e.* I = ABC. From step I.3 and I.5 of this section, we retain the treatment combinations corresponding to the equation $x_1 + x_2 + x_3 = 0$. We have

Design 3.3

	A	**B**	**C**	**D**	**E**
(1)	0	0	0	0	0
ab	1	1	0	0	0

ac	1	0	1	0	0
bc	0	1	1	0	0
d	0	0	0	1	0
abd	1	1	0	1	0
acd	1	0	1	1	0
bcd	0	1	1	1	0
e	0	0	0	0	1
abe	1	1	0	0	1
ace	1	0	1	0	1
bce	0	1	1	0	1
de	0	0	0	1	1
abde	1	1	0	1	1
acde	1	0	1	1	1
bcde	0	1	1	1	1

For one-fourth replicate, use another defining contrast CDE and solve for equation $x_3 + x_4 + x_5 = 0$. Their generalized contrast is ABDE. Finally we have
I = ABC = CDE = ABDE and the design ¼ (2^5) is given as Design 3.4.

Design 3.4

	A	**B**	**C**	**D**	**E**
(1)	0	0	0	0	0
ab	1	1	0	0	0
acd	1	0	1	1	0
bcd	0	1	1	1	0
ace	1	0	1	0	1
bce	0	1	1	0	1
de	0	0	0	1	1
abde	1	1	0	1	1

As per II.4 of this section, change all 0's by -1 say the **Design *D***

Design *D*

	A	**B**	**C**	**D**	**E**
(1)	-1	-1	-1	-1	-1
ab	1	1	-1	-1	-1

acd	1	-1	1	1	-1
bcd	-1	1	1	1	-1
ace	1	-1	1	-1	1
bce	-1	1	1	-1	1
de	-1	-1	-1	1	1
abde	1	1	-1	1	1

As per II.5 of this section, generate the design $\mathbf{D'}$ by taking the complement foldover of **D** by replacing all 1 by -1 and -1 by 1.

Design D'

	A	B	C	D	E
abcde	1	1	1	1	1
cde	-1	-1	1	1	1
be	-1	1	-1	-1	1
ae	1	-1	-1	-1	1
bd	-1	1	-1	1	-1
ad	1	-1	-1	1	-1
abc	1	1	1	-1	-1
c	-1	-1	1	-1	-1

As per II.6 of this section, combine the design ***D*** and $\mathbf{D'}$. This is desired Design ***D****.

Design *D*

	A	B	C	D	E
1	-1	-1	-1	-1	-1
ab	1	1	-1	-1	-1
acd	1	-1	1	1	-1
bcd	-1	1	1	1	-1
ace	1	-1	1	-1	1
bce	-1	1	1	-1	1
de	-1	-1	-1	1	1
abde	1	1	-1	1	1
abcde	1	1	1	1	1
cde	-1	-1	1	1	1
be	-1	1	-1	-1	1
ae	1	-1	-1	-1	1
bd	-1	1	-1	1	-1
ad	1	-1	-1	1	-1
abc	1	1	1	-1	-1
c	-1	-1	1	-1	-1

This is the desired design ***D****. In this design treatment combinations of Defining contrasts ABC and CDE are retained and I = ABDE is still defining contrast.

Thus the design ***D**** is the design that is $\frac{1}{2}(2^5)$ in which the defining contrast is I = ABDE and all the main effects A, B, C, D and E are linear trend-free. Now we examine whether all main effects of the design $\boldsymbol{D}^*$ are orthogonal to linear-trend by the following procedure:

	A	**B**	**C**	**D**	**E**	**T**
	-1	-1	-1	-1	-1	-15
	1	1	-1	-1	-1	-13
	1	-1	1	-1	-1	-11
	-1	1	1	-1	-1	-9
	-1	-1	-1	1	1	-7
	1	1	-1	1	1	-5
	1	-1	1	1	1	-3
X=	-1	1	1	1	1	-1
	1	1	1	1	1	1
	-1	-1	1	1	1	3
	-1	1	-1	1	1	5
	1	-1	-1	1	1	7
	1	1	1	-1	-1	9
	-1	-1	1	-1	-1	11
	-1	1	-1	-1	-1	13
	1	-1	-1	-1	-1	15

	A	**B**	**C**	**D**	**E**
	15	15	15	15	15
	-13	-13	13	13	13
	-11	11	-11	11	11
	9	-9	-9	9	9
	7	7	7	-7	-7
	-5	-5	5	-5	-5
	-3	3	-3	-3	-3
X′T =	1	-1	-1	-1	-1
	1	1	1	1	1
	-3	-3	3	3	3
	-5	5	-5	5	5
	7	-7	-7	7	7
	9	9	9	-9	-9
	-11	-11	11	-11	-11
	-13	13	-13	-13	-13
	15	-15	-15	-15	-15
Sum	**0**	**0**	**0**	**0**	**0**

Further to identify the two factor interactions are linear trend-free/ linear trend- free, we proceed as per steps III.1 to III.5 of section 2.3. The two factor interactions AB, AD, AE, BC, BD, BE, DE are linear trend-free and AC is nearly linear trend-free.

The algorithm given in Section 3.3 was translated in Microsoft Visual C++ program. The listing of the program is given in Appendix 3.1. From this program, one can obtain the desired fractional factorial experiment *k* (≥5). The catalogue of the obtained fractional factorial experiment (for *k* = 5, 6, 7 and 8), with varying fractions, that are linear trend-free for main effects is given in Appendix 3.2. The identified two factor interactions that are linear trend-free are given along with the design.

Appendix-3.1

Computer program in Microsoft Visual C++ for obtaining fractional factorial experiment that are linear trend-free for main effects

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
static int a[1024][10], b[10], b1[10], b2[10],t[1024];
int a1[1024][10], a2[1024][10], a3[1024][10],c1[1024][10],d1[1024][10];
void main()
{
        FILE *fp;
        fp=fopen("d:\obj2.txt","a+");
        int
i,d,j=0,h=0,q,l,m,m1,n,sum,y=0,z=0,n1,j1,fr=0,k=0,n2,k1=0,n3,k2=0,k3=0,c,c5;
printf("Enter the number of factors K = ");
scanf("%d",&n);
if(fp!=NULL){
m=pow(2,n);
for(i=0;i<n;i++)
        y=y+i;
for(j=0;j<n;j++)
{
  if(j==0)
  {
   for(i=0;i<m;i++)
    {
     if(i%2==0)
     a[i][j]=0;
     else
     a[i][j]=1;
    }
   }
  else
  {
    for(i=0;i<m;)
    {
        if(h%2==0)
        {
          for(l=0;l<pow(2,j);l++)
           {
           a[i][j]=0;
           i++;
           }
        }
```

```
        else
        {
         for(l=0;l<pow(2,j);l++)
          {
           a[i][j]=1;
           i++;
          }
        }
   h++;
   }
}
h=0;
}
/**************************/
printf("Enter 1 for half fraction");
printf("Enter 2 for quarter fraction");
printf("Enter 3 for 1/8 fraction");
scanf("%d",&fr);
/*****************/
if(fr==1)
{
        printf("Enter the number of factors for interaction");
        scanf("%d",&n1);
        printf("Enter interaction");
        for(i=0;i<n1;i++)
        {
                scanf("%d",&b[i]);
        }

        for (i=0;i<2*m;i++)
        {
        q=(-(m-(i+1)));
        t[i/2]=q;
        fprintf(fp,"%3d",t[i/2]);
        fprintf(fp,"\n");
        i++;
                }
        for(i=0;i<m;i++)
   {
                sum=0;
   for(j1=0;j1<n1;j1++)
        {
                sum=sum+a[i][b[j1]];
        }
                if((sum%2)==0)
                        {
```

```
                                        for(j=0;j<n;j++)
                                                {
                                                a1[k][j]=a[i][j];

                                                }
                                                k++;
                                }
                }
for(i=0;i<n1;i++)
{
                fprintf(fp,"\n%3d",b[i]);
}
          for(k=0;k<m/2;k++)
                {
                for(j=0;j<n;j++)
                        {
                                fprintf(fp,"%3d",a1[k][j]);
                                fprintf(fp,"\t");
                        }
                   fprintf(fp,"\n");
                }

          for(k=0;k<m/2;k++)
                {
                for(j=0;j<n;j++)
                        {
                                fprintf(fp,"%3d",a1[k][j]);
                                fprintf(fp,"\t");
                        }
                   fprintf(fp,"\n");
                }

          for(k=0;k<m/2;k++)
                {

                for(j=0;j<n;j++)
                        {
                        if(a1[k][j]==0)
                                {
                                d=1;
                                fprintf(fp,"%3d",d);
                                fprintf(fp,"\t");
                                }
                        else
                        {
                                d=0;
```

```
            fprintf(fp,"%3d",d);
            fprintf(fp,"\t");
        }
        }
     fprintf(fp,"\n");
    }
for(k=0;k<m/2;k++)
    {
    for(j=0;j<n;j++)
        {
        if(a1[k][j]==0)
            {
            a1[k][j]=-1;
            c1[k][j]=a1[k][j];
            c=c1[k][j];
            fprintf(fp,"%3d",c);
            fprintf(fp,"\t");
            }
        else
        {
            fprintf(fp,"%3d",a1[k][j]);
            fprintf(fp,"\t");
        }
        }
     fprintf(fp,"\n");
    }
for(k=0;k<m/2;k++)
    {
    for(j=0;j<n;j++)
        {
        if(a1[k][j]==0)
            {
            a1[k][j]=-1;
            c1[k][j]=a1[k][j];
            fprintf(fp,"%3d",a1[k][j]);
            fprintf(fp,"\t");
            }
        else
        {
            c1[k][j]=a1[k][j];
            fprintf(fp,"%3d",a1[k][j]);
            fprintf(fp,"\t");
        }
        }
     fprintf(fp,"\n");
    }
```

```
        k3=m/2;
        for(k=0;k<m/2;k++)
                {

                for(j=0;j<n;j++)
                        {
                        if(a1[k][j]==0)
                                {
                                a1[k][j]=-1;
                                c=-1*a1[k][j];
                                c1[k3][j]=c;
                                fprintf(fp,"%3d",c);
                                fprintf(fp,"\t");
                                }
                        else
                        {
                                c=-1*a1[k][j];
                                c1[k3][j]=c;
                                fprintf(fp,"%3d",c);
                                fprintf(fp,"\t");
                        }
                        }
                k3++;
                   fprintf(fp,"\n");
                }

        for(k=0;k<m;k++)
        {

                for(j=0;j<n;j++)
                {
                        c1[k][j]=t[k]*c1[k][j];
                        fprintf(fp,"%3d",c1[k][j]);
                        fprintf(fp,"\t");
                }
                fprintf(fp,"\n");
        }

sum=0;
for(k=0;k<n;k++)
   {
        for(i=0;i<m;i++)
        {
                sum=sum+c1[i][k];
        }
        fprintf(fp,"%3d",sum);
```

```
          sum=0;
          fprintf(fp,"\t");
   }
}
/**************/
if(fr==2)
{
          printf("Enter the number of factors for first interaction");
          scanf("%d",&n1);
          printf("Enter interaction");
          for(i=0;i<n1;i++)
          {
                    scanf("%d",&b[i]);
          }
          printf("Enter the number of factors for second interaction");
          scanf("%d",&n2);
          printf("Enter interaction");
          for(i=0;i<n2;i++)
          {
                    scanf("%d",&b1[i]);
          }

          m1=m/2;
          for (i=0;i<2*m1;i++)
          {
          q=(-(m1-(i+1)));
          t[i/2]=q;
          fprintf(fp,"%3d",t[i/2]);
          fprintf(fp,"\n");
          i++;
                    }

          for(i=0;i<m;i++)
   {
                    sum=0;
   for(j1=0;j1<n1;j1++)
          {
                    sum=sum+a[i][b[j1]];
          }
                    if((sum%2)==0)
                              {
                                        for(j=0;j<n;j++)
                                                  {
                                                  a1[k][j]=a[i][j];

                                                  }
```

```
                    k++;
            }
    }
for(i=0;i<n1;i++)
{
        fprintf(fp,"\n%3d",b[i]);
}
    for(k=0;k<m/2;k++)
        {
        for(j=0;j<n;j++)
            {
            fprintf(fp,"%3d",a1[k][j]);
            fprintf(fp,"\t");
            }
           fprintf(fp,"\n");
        }
    for(k=0;k<m/2;k++)
   {
        sum=0;
   for(j1=0;j1<n2;j1++)
    {
        sum=sum+a1[k][b1[j1]];
    }
        if((sum%2)==0)
            {
                for(j=0;j<n;j++)
                    {
                    a2[k1][j]=a1[k][j];

                    }
                    k1++;
            }
    }
fprintf(fp,"\nb1\n");
for(i=0;i<n2;i++)
{
        fprintf(fp,"\n%3d",b1[i]);
}
    for(k1=0;k1<m/4;k1++)
        {
        for(j=0;j<n;j++)
            {
                fprintf(fp,"%3d",a2[k1][j]);
                fprintf(fp,"\t");
            }
               fprintf(fp,"\n");
```

```
        }
for(k1=0;k1<m/4;k1++)
        {
        for(j=0;j<n;j++)
            {
                fprintf(fp,"%3d",a2[k1][j]);
                fprintf(fp,"\t");
            }
             fprintf(fp,"\n");
        }
for(k1=0;k1<m/4;k1++)
        {
        for(j=0;j<n;j++)
            {
            if(a2[k1][j]==0)
            {
                d=1;
                fprintf(fp,"%3d",d);
                fprintf(fp,"\t");
            }
            else
            {
                d=0;
                fprintf(fp,"%3d",d);
                fprintf(fp,"\t");
            }
        }
             fprintf(fp,"\n");
        }

    for(k1=0;k1<m/4;k1++)
        {
        for(j=0;j<n;j++)
            {
            if(a2[k1][j]==0)
                {
                a2[k1][j]=-1;
                fprintf(fp,"%3d",a2[k1][j]);
                fprintf(fp,"\t");
                }
            else
            {
                fprintf(fp,"%3d",a2[k1][j]);
                fprintf(fp,"\t");
            }
        }
```

```
                    fprintf(fp,"\n");
            }

for(k1=0;k1<m/4;k1++)
            {
            for(j=0;j<n;j++)
                        {
                        if(a2[k1][j]==0)
                                    {
                                    a2[k1][j]=-1;
                                    c1[k1][j]=a2[k1][j];
                                    fprintf(fp,"%3d",a2[k1][j]);
                                    fprintf(fp,"\t");
                                    }
                        else
                        {
                                    c1[k1][j]=a2[k1][j];
                                    fprintf(fp,"%3d",a2[k1][j]);
                                    fprintf(fp,"\t");
                        }
            }
                    fprintf(fp,"\n");
            }
            k3=m/4;
for(k1=0;k1<m/4;k1++)
            {
            for(j=0;j<n;j++)
                        {
                        if(a2[k1][j]==0)
                                    {
                                    a2[k1][j]=-1;
                                    c=a2[k1][j]*-1;
                                    c1[k3][j]=c;
                                    fprintf(fp,"%3d",c);
                                    fprintf(fp,"\t");
                                    }
                        else
                        {
                                    c=a2[k1][j]*-1;
                                    c1[k3][j]=c;
                                    fprintf(fp,"%3d",c);
                                    fprintf(fp,"\t");
                        }
            }k3++;
                    fprintf(fp,"\n");
            }
```

```
        for(k=0;k<m/2;k++)
        {

                for(j=0;j<n;j++)
                {
                        c1[k][j]=t[k]*c1[k][j];
                        fprintf(fp,"%3d",c1[k][j]);
                        fprintf(fp,"\t");
                }
                fprintf(fp,"\n");
        }

sum=0;
for(k=0;k<n;k++)
   {
        for(i=0;i<m/2;i++)
        {
                sum=sum+c1[i][k];

        }
        fprintf(fp,"%3d",sum);
        sum=0;
        fprintf(fp,"\t");
   }

}
if(fr==3)
{
        printf("Enter the number of factors for first interaction");
        scanf("%d",&n1);
        printf("Enter interaction");
        for(i=0;i<n1;i++)
        {
                scanf("%d",&b[i]);
        }
        printf("Enter the number of factors for second interaction");
        scanf("%d",&n2);
        printf("Enter interaction");
        for(i=0;i<n2;i++)
        {
                scanf("%d",&b1[i]);
        }
        printf("Enter the number of factors for third interaction");
        scanf("%d",&n3);
        printf("Enter interaction");
```

```
          for(i=0;i<n3;i++)
          {
                   scanf("%d",&b2[i]);
          }


          m1=m/4;
          for (i=0;i<2*m1;i++)
          {
          q=(-(m1-(i+1)));
          t[i/2]=q;
          fprintf(fp,"%3d",t[i/2]);
          fprintf(fp,"\n");
          i++;
                   }
                   for(i=0;i<m;i++)
    {
                   sum=0;
    for(j1=0;j1<n1;j1++)
          {
                   sum=sum+a[i][b[j1]];
          }
                   if((sum%2)==0)
                            {
                                     for(j=0;j<n;j++)
                                              {
                                              a1[k][j]=a[i][j];

                                              }
                                              k++;
                            }
          }
fprintf(fp,"\nb\n");
          for(i=0;i<n1;i++)
{
                   fprintf(fp,"\n%3d",b[i]);
}
          for(k=0;k<m/2;k++)
                   {
                   for(j=0;j<n;j++)
                            {
                            fprintf(fp,"%3d",a1[k][j]);
                            fprintf(fp,"\t");
                            }
                      fprintf(fp,"\n");
                   }
```

```
        for(k=0;k<m/2;k++)
  {
                sum=0;
  for(j1=0;j1<n2;j1++)
        {
                sum=sum+a1[k][b1[j1]];
        }
                if((sum%2)==0)
                        {
                                for(j=0;j<n;j++)
                                        {
                                        a2[k1][j]=a1[k][j];

                                        }
                                        k1++;
                        }
        }

fprintf(fp,"\nb1\n");
for(i=0;i<n2;i++)
{
                fprintf(fp,"\n%3d",b1[i]);
}
        for(k1=0;k1<m/4;k1++)
                {
                for(j=0;j<n;j++)
                        {
                        fprintf(fp,"%3d",a2[k1][j]);
                        fprintf(fp,"\t");
                        }
                  fprintf(fp,"\n");
                }

        for(k1=0;k1<m/4;k1++)
  {
                sum=0;
  for(j1=0;j1<n3;j1++)
        {
                sum=sum+a2[k1][b2[j1]];
        }
                if((sum%2)==0)
                        {
                                for(j=0;j<n;j++)
                                        {
                                        a3[k2][j]=a2[k1][j];
```

```
                                        }
                                        k2++;
                                }
                }

fprintf(fp,"\nb2\n");
for(i=0;i<n3;i++)
{
                fprintf(fp,"\n%3d",b2[i]);
}

        for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                                fprintf(fp,"%3d",a3[k2][j]);
                                fprintf(fp,"\t");
                        }

                   fprintf(fp,"\n");
                }
        for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                                fprintf(fp,"%3d",a3[k2][j]);
                                fprintf(fp,"\t");
                        }
                   fprintf(fp,"\n");
                }
for(k2=0;k2<m/8;k2++)
                {
                for(j=0;j<n;j++)
                        {
                        if(a3[k2][j]==0)
                        {
                                d=1;
                                fprintf(fp,"%3d",d);
                                fprintf(fp,"\t");
                        }
                        else
                        {
                                d=0;
                                fprintf(fp,"%3d",d);
                                fprintf(fp,"\t");
                        }
```

```
        }
          fprintf(fp,"\n");
        }
    for(k2=0;k2<m/8;k2++)
        {
        for(j=0;j<n;j++)
            {
            if(a3[k2][j]==0)
                {
                a3[k2][j]=-1;
                fprintf(fp,"%3d",a3[k2][j]);
                fprintf(fp,"\t");
                }
            else
            {
                fprintf(fp,"%3d",a3[k2][j]);
                fprintf(fp,"\t");
            }
            }

          fprintf(fp,"\n");
        }
for(k2=0;k2<m/8;k2++)
        {
        for(j=0;j<n;j++)
            {
            if(a3[k2][j]==0)
                {
                a3[k2][j]=-1;
                c1[k2][j]=a3[k2][j];
                fprintf(fp,"%3d",a3[k2][j]);
                fprintf(fp,"\t");
                }
            else
            {
                c1[k2][j]=a3[k2][j];
                fprintf(fp,"%3d",a3[k2][j]);
                fprintf(fp,"\t");
            }
            }

          fprintf(fp,"\n");
        }
    k3=m/8;
    for(k2=0;k2<m/8;k2++)
        {
```

```
                        for(j=0;j<n;j++)
                                {
                                if(a3[k2][j]==0)
                                        {
                                        a3[k2][j]=-1;
                                        c=a3[k][j]*-1;
                                        c1[k3][j]=c;
                                        fprintf(fp,"%3d",c);
                                        fprintf(fp,"\t");
                                        }
                                else
                                {
                                        c=a3[k2][j]*-1;
                                        c1[k3][j]=c;
                                        fprintf(fp,"%3d",c);
                                        fprintf(fp,"\t");
                                }
                                }k3++;

                          fprintf(fp,"\n");
                        }

                for(k=0;k<m/4;k++)
                {

                        for(j=0;j<n;j++)
                        {
                                c1[k][j]=t[k]*c1[k][j];
                                fprintf(fp,"%3d",c1[k][j]);
                                fprintf(fp,"\t");
                        }
                        fprintf(fp,"\n");
                }

        sum=0;
        for(k=0;k<n;k++)
          {
                for(i=0;i<m/4;i++)
                {
                        sum=sum+c1[i][k];

                }
                fprintf(fp,"%3d",sum);
                sum=0;
                fprintf(fp,"\t");
          }
```

```
}
fprintf(fp,"\nfr=%d ",fr);
}
fclose(fp);
getch();
}
```

Appendix-3.2

Catalogue of fractional factorial experiments that are linear trend-free for main effects (for *k* = 5, 6, 7 and 8) and some of two and three factor interactions

By using the algorithm developed in section 3.1, computer program in Microsoft Visual C++ is run for obtaining the desired fractional factorial experiment *k* (≥5). The catalogue of the obtained fractional factorial experiment (for *k* = 5, 6, 7, 8).

½ (2^5) Experiment, I = ABDE

	A	B	C	D	E
(1)	-1	-1	-1	-1	-1
ab	1	1	-1	-1	-1
acd	1	-1	1	1	-1
acd	-1	1	1	1	-1
ace	1	-1	1	-1	1
bce	-1	1	1	-1	1
de	-1	-1	-1	1	1
abde	1	1	-1	1	1
abcde	1	1	1	1	1
cde	-1	-1	1	1	1
be	-1	1	-1	-1	1
ae	1	-1	-1	-1	1
bd	-1	1	-1	1	-1
ad	1	-1	-1	1	-1
abc	1	1	1	-1	-1
c	-1	-1	1	-1	-1

LTF: AB, AD, AE, BC, BD, BE, DE; NTLF: AC

½ (2^5) Experiment, I = ABCD

	A	B	C	D	E
(1)	-1	-1	-1	-1	-1
ab	1	1	-1	-1	-1
cd	-1	-1	1	1	-1
abcd	1	1	1	1	-1
ace	1	-1	1	-1	1
bce	-1	1	1	-1	1
ade	1	-1	-1	1	1
bde	-1	1	-1	1	1
abcde	1	1	1	1	1
cde	-1	-1	1	1	1
abe	1	1	-1	-1	1
e	-1	-1	-1	-1	1

bd	-1	1	-1	1	-1
ad	1	-1	-1	1	-1
bc	-1	1	1	-1	-1
ac	1	-1	1	-1	-1

LTF: AC, AD, BC, BD, BE, DE; NTLF: AE

½ (2^6) Experiment I = ABCD

	A	**B**	**C**	**D**	**E**	**F**
(1)	-1	-1	-1	-1	-1	-1
ab	1	1	-1	-1	-1	-1
ac	1	-1	1	-1	-1	-1
bc	-1	1	1	-1	-1	-1
ade	1	-1	-1	1	1	-1
bde	-1	1	-1	1	1	-1
cde	-1	-1	1	1	1	-1
abcde	1	1	1	1	1	-1
adf	1	-1	-1	1	-1	1
bdf	-1	1	-1	1	-1	1
cdf	-1	-1	1	1	-1	1
abcdf	1	1	1	1	-1	1
ef	-1	-1	-1	-1	1	1
abef	1	1	-1	-1	1	1
acef	1	-1	1	-1	1	1
bcef	-1	1	1	-1	1	1
abcdef	1	1	1	1	1	1
cdef	-1	-1	1	1	1	1
bdef	-1	1	-1	1	1	1
adef	1	-1	-1	1	1	1
bcf	-1	1	1	-1	-1	1
acf	1	-1	1	-1	-1	1
abf	1	1	-1	-1	-1	1
f	-1	-1	-1	-1	-1	1
bce	-1	1	1	-1	1	-1
ace	1	-1	1	-1	1	-1
abe	1	1	-1	-1	1	-1
e	-1	-1	-1	-1	1	-1

abcd	1	1	1	1	-1	-1
cd	-1	-1	1	1	-1	-1
bd	-1	1	-1	1	-1	-1
ad	1	-1	-1	1	-1	-1

LTF: All two and three factor interactions are except DE and DF

.

½ (2^6) Experiment, I = ABCDEF

	A	B	C	D	E	F
(1)	-1	-1	-1	-1	-1	-1
ab	1	1	-1	-1	-1	-1
ac	1	-1	1	-1	-1	-1
bc	-1	1	1	-1	-1	-1
de	-1	-1	-1	1	1	-1
abde	1	1	-1	1	1	-1
acde	1	-1	1	1	1	-1
bcde	-1	1	1	1	1	-1
df	-1	-1	-1	1	-1	1
abdf	1	1	-1	1	-1	1
acdf	1	-1	1	1	-1	1
bcdf	-1	1	1	1	-1	1
ef	-1	-1	-1	-1	1	1
abef	1	1	-1	-1	1	1
acef	1	-1	1	-1	1	1
bcef	-1	1	1	-1	1	1
abcdef	1	1	1	1	1	1
cdef	-1	-1	1	1	1	1
bdef	-1	1	-1	1	1	1
adef	1	-1	-1	1	1	1
abcf	1	1	1	-1	-1	1
cf	-1	-1	1	-1	-1	1
af	-1	1	-1	-1	-1	1
af	1	-1	-1	-1	-1	1
abce	1	1	1	-1	1	-1
ce	-1	-1	1	-1	1	-1
be	-1	1	-1	-1	1	-1
ae	1	-1	-1	-1	1	-1
abcd	1	1	1	1	-1	-1
cd	-1	-1	1	1	-1	-1
bd	-1	1	-1	1	-1	-1
ad	1	-1	-1	1	-1	-1

LTF: All two and three factor interactions are except DE and DF

½ (2^7) Experiment, I = ABCDFG

	A	B	C	D	E	F	G		A	B	C	D	E	F	G
(1)	-1	-1	-1	-1	-1	-1	-1	**abcdefg**	1	1	1	1	1	1	1
ab	1	1	-1	-1	-1	-1	-1	**cdefg**	-1	-1	1	1	1	1	1
ac	1	-1	1	-1	-1	-1	-1	**bdefg**	-1	1	-1	1	1	1	1
bc	-1	1	1	-1	-1	-1	-1	**adefg**	1	-1	-1	1	1	1	1
ad	1	-1	-1	1	-1	-1	-1	**bcefg**	-1	1	1	-1	1	1	1
bd	-1	1	-1	1	-1	-1	-1	**acefg**	1	-1	1	-1	1	1	1
cd	-1	-1	1	1	-1	-1	-1	**abefg**	1	1	-1	-1	1	1	1
abcd	1	1	1	1	-1	-1	-1	**efg**	-1	-1	-1	-1	1	1	1
aef	1	-1	-1	-1	1	1	-1	**bcdg**	-1	1	1	1	-1	-1	1
bef	-1	1	-1	-1	1	1	-1	**acdg**	1	-1	1	1	-1	-1	1
cef	-1	-1	1	-1	1	1	-1	**abdg**	1	1	-1	1	-1	-1	1
abcef	1	1	1	-1	1	1	-1	**dg**	-1	-1	-1	1	-1	-1	1
def	-1	-1	-1	1	1	1	-1	**abcg**	1	1	1	-1	-1	-1	1
abdef	1	1	-1	1	1	1	-1	**cg**	-1	-1	1	-1	-1	-1	1
acdef	1	-1	1	1	1	1	-1	**bg**	-1	1	-1	-1	-1	-1	1
bcdef	-1	1	1	1	1	1	-1	**ag**	1	-1	-1	-1	-1	-1	1
aeg	1	-1	-1	-1	1	-1	1	**bcdf**	-1	1	1	1	-1	1	-1
beg	-1	1	-1	-1	1	-1	1	**acdf**	1	-1	1	1	-1	1	-1
ceg	-1	-1	1	-1	1	-1	1	**abdf**	1	1	-1	1	-1	1	-1
abceg	1	1	1	-1	1	-1	1	**df**	-1	-1	-1	1	-1	1	-1
deg	-1	-1	-1	1	1	-1	1	**abcf**	1	1	1	-1	-1	1	-1
abdeg	1	1	-1	1	1	-1	1	**cf**	-1	-1	1	-1	-1	1	-1
acdeg	1	-1	1	1	1	-1	1	**bf**	-1	1	-1	-1	-1	1	-1
bcdeg	-1	1	1	1	1	-1	1	**af**	1	-1	-1	-1	-1	1	-1
fg	-1	-1	-1	-1	-1	1	1	**abcde**	1	1	1	1	1	-1	-1
abfg	1	1	-1	-1	-1	1	1	**cde**	-1	-1	1	1	1	-1	-1
acfg	1	-1	1	-1	-1	1	1	**bde**	-1	1	-1	1	1	-1	-1
bcfg	-1	1	1	-1	-1	1	1	**ade**	1	-1	-1	1	1	-1	-1
adfg	1	-1	-1	1	-1	1	1	**bce**	-1	1	1	-1	1	-1	-1
bdfg	-1	1	-1	1	-1	1	1	**ace**	1	-1	1	-1	1	-1	-1
cdfg	-1	-1	1	1	-1	1	1	**abe**	1	1	-1	-1	1	-1	-1
abcdfg	1	1	1	1	-1	1	1	**e**	-1	-1	-1	-1	1	-1	-1

LTF: All two and three factor interactions except EF and EG

¼ (2^8) factorial I = ABDE= ABCFGH = CDEFGH

	A	B	C	D	E	F	G	H		A	B	C	D	E	F	G	H
(1)	-1	-1	-1	-1	-1	-1	-1	-1	**abcdefgh**	1	1	1	1	1	1	1	1
ab	1	1	-1	-1	-1	-1	-1	-1	**cdefgh**	-1	-1	1	1	1	1	1	1
acd	1	-1	1	1	-1	-1	-1	-1	**befgh**	-1	1	-1	-1	1	1	1	1
bcd	-1	1	1	1	-1	-1	-1	-1	**aefgh**	1	-1	-1	-1	1	1	1	1
ace	1	-1	1	-1	1	-1	-1	-1	**bdfgh**	-1	1	-1	1	-1	1	1	1
bce	-1	1	1	-1	1	-1	-1	-1	**adfgh**	1	-1	-1	1	-1	1	1	1
de	-1	-1	-1	1	1	-1	-1	-1	**abcfgh**	1	1	1	-1	-1	1	1	1
abde	1	1	-1	1	1	-1	-1	-1	**cfgh**	-1	-1	1	-1	-1	1	1	1
fg	-1	-1	-1	-1	-1	1	1	-1	**abcdeh**	1	1	1	1	1	-1	-1	1

abfg	1	1	-1	-1	-1	1	1	-1	**cdeh**	-1	-1	1	1	1	-1	-1	1
acdfg	1	-1	1	1	-1	1	1	-1	**beh**	-1	1	-1	-1	1	-1	-1	1
bcdfg	-1	1	1	1	-1	1	1	-1	**aeh**	1	-1	-1	-1	1	-1	-1	1
acefg	1	-1	1	-1	1	1	1	-1	**bdh**	-1	1	-1	1	-1	-1	-1	1
bcefg	-1	1	1	-1	1	1	1	-1	**adh**	1	-1	-1	1	-1	-1	-1	1
defg	-1	-1	-1	1	1	1	1	-1	**abch**	1	1	1	-1	-1	-1	-1	1
abdefg	1	1	-1	1	1	1	1	-1	**ch**	-1	-1	1	-1	-1	-1	-1	1
fh	-1	-1	-1	-1	-1	1	-1	1	**abcdeg**	1	1	1	1	1	-1	1	-1
abfh	1	1	-1	-1	-1	1	-1	1	**cdeg**	-1	-1	1	1	1	-1	1	-1
acdfh	1	-1	1	1	-1	1	-1	1	**beg**	-1	1	-1	-1	1	-1	1	-1
bcdfh	-1	1	1	1	-1	1	-1	1	**aeg**	1	-1	-1	-1	1	-1	1	-1
acefh	1	-1	1	-1	1	1	-1	1	**bdg**	-1	1	-1	1	-1	-1	1	-1
bcefh	-1	1	1	-1	1	1	-1	1	**adg**	1	-1	-1	1	-1	-1	1	-1
defh	-1	-1	-1	1	1	1	-1	1	**abcg**	1	1	1	-1	-1	-1	1	-1
abdefh	1	1	-1	1	1	1	-1	1	**dg**	-1	-1	1	-1	-1	-1	1	-1
gh	-1	-1	-1	-1	-1	-1	1	1	**abcdef**	1	1	1	1	1	1	-1	-1
abgh	1	1	-1	-1	-1	-1	1	1	**cdef**	-1	-1	1	1	1	1	-1	-1
acdgh	1	-1	1	1	-1	-1	1	1	**bef**	-1	1	-1	-1	1	1	-1	-1
bcdgh	-1	1	1	1	-1	-1	1	1	**aef**	1	-1	-1	-1	1	1	-1	-1
acegh	1	-1	1	-1	1	-1	1	1	**bdf**	-1	1	-1	1	-1	1	-1	-1
acegh	-1	1	1	-1	1	-1	1	1	**adf**	1	-1	-1	1	-1	1	-1	-1
degh	-1	-1	-1	1	1	-1	1	1	**abcf**	1	1	1	-1	-1	1	-1	-1
abdegh	1	1	-1	1	1	-1	1	1	**cf**	-1	-1	1	-1	-1	1	-1	-1

LTF: All two and three factor interactions except CD, CE, FG and FH; NTLF: AC

¼ (2^8) Experiment, I = ABCDEF = DEGH = ABCFGH

	A	B	C	D	E	F	G	H		A	B	C	D	E	F	G	H
(1)	-1	-1	-1	-1	-1	-1	-1	-1	**abcdefgh**	1	1	1	1	1	1	1	1
ab	1	1	-1	-1	-1	-1	-1	-1	**cdefgh**	-1	-1	1	1	1	1	1	1
ac	1	-1	1	-1	-1	-1	-1	-1	**bdefgh**	-1	1	-1	1	1	1	1	1
bc	-1	1	1	-1	-1	-1	-1	-1	**adefgh**	1	-1	-1	1	1	1	1	1
de	-1	-1	-1	1	1	-1	-1	-1	**abcfgh**	1	1	1	-1	-1	1	1	1
abde	1	1	-1	1	1	-1	-1	-1	**cfgh**	-1	-1	1	-1	-1	1	1	1
acde	1	-1	1	1	1	-1	-1	-1	**bfgh**	-1	1	-1	-1	-1	1	1	1
bcde	-1	1	1	1	1	-1	-1	-1	**afgh**	1	-1	-1	-1	-1	1	1	1
dfg	-1	-1	-1	1	-1	1	1	-1	**abceh**	1	1	1	-1	1	-1	-1	1
abdfg	1	1	-1	1	-1	1	1	-1	**ceh**	-1	-1	1	-1	1	-1	-1	1
acdfg	1	-1	1	1	-1	1	1	-1	**beh**	-1	1	-1	-1	1	-1	-1	1
bcdfg	-1	1	1	1	-1	1	1	-1	**aeh**	1	-1	-1	-1	1	-1	-1	1
efg	-1	-1	-1	-1	1	1	1	-1	**abcdh**	1	1	1	1	-1	-1	-1	1
abefg	1	1	-1	-1	1	1	1	-1	**cdh**	-1	-1	1	1	-1	-1	-1	1
acefg	1	-1	1	-1	1	1	1	-1	**bdh**	-1	1	-1	1	-1	-1	-1	1
bcefg	-1	1	1	-1	1	1	1	-1	**adh**	1	-1	-1	1	-1	-1	-1	1
dfh	-1	-1	-1	1	-1	1	-1	1	**abceg**	1	1	1	-1	1	-1	1	-1
abdfh	1	1	-1	1	-1	1	-1	1	**ceg**	-1	-1	1	-1	1	-1	1	-1
acdfh	1	-1	1	1	-1	1	-1	1	**beg**	-1	1	-1	-1	1	-1	1	-1
bcdfh	-1	1	1	1	-1	1	-1	1	**aeg**	1	-1	-1	-1	1	-1	1	-1
efh	-1	-1	-1	-1	1	1	-1	1	**abcdg**	1	1	1	1	-1	-1	1	-1
abefh	1	1	-1	-1	1	1	-1	1	**cdg**	-1	-1	1	1	-1	-1	1	-1

acefh	1	-1	1	-1	1	1	-1	1	**bdg**	-1	1	-1	1	-1	-1	1	-1
bcefh	-1	1	1	-1	1	1	-1	1	**adg**	1	-1	-1	1	-1	-1	1	-1
gh	-1	-1	-1	-1	-1	-1	1	1	**abcdef**	1	1	1	1	1	1	-1	-1
abgh	1	1	-1	-1	-1	-1	1	1	**cdef**	-1	-1	1	1	1	1	-1	-1
acgh	1	-1	1	-1	-1	-1	1	1	**bdef**	-1	1	-1	1	1	1	-1	-1
bcgh	-1	1	1	-1	-1	-1	1	1	**adef**	1	-1	-1	1	1	1	-1	-1
degh	-1	-1	-1	1	1	-1	1	1	**abcf**	1	1	1	-1	-1	1	-1	-1
abdegh	1	1	-1	1	1	-1	1	1	**cf**	-1	-1	1	-1	-1	1	-1	-1
acdegh	1	-1	1	1	1	-1	1	1	**bf**	-1	1	-1	-1	-1	1	-1	-1
bcdegh	-1	1	1	1	1	-1	1	1	**af**	1	-1	-1	-1	-1	1	-1	-1

LTF: All two and three factor interactions except DF, FG and FH; NTLF: AC

Chapter IV

DEVELOPMENT OF LINEAR TREND-FREE RESPONSE SURFACE DESIGNS

4.1 Introduction

Data from experiments with levels or level combinations of one or more factors as treatments are normally investigated to compare level effects of the factors and also their interactions. Though such investigations are useful to have objective assessment of the effect of levels tried in the experiment, this seems to have inadequate when the data are quantitative in nature. The analysis as such does not give any information regarding the possible effects of the intervening levels of the factors or their combinations, i.e. one is not able to interpolate the responses at the treatment combinations not tried in the experiment. In such cases, it is more realistic and informative to carry out investigations with the twofold purposes:

1. To determine and quantify the relationship between the values of one or more measurable response variable(s) and the setting of a group of experimental factors presumed to affect the response(s).

2. To find the settings of the experimental factors that produce the best value or the best set of values of the response(s).

If all the factors are quantitative in nature, it is proper to think the response as a function of the factor levels and data from quantitative factorial experiments can be used to fit the response surfaces over the region of interest. Response surfaces besides inferring about the twin purposes can provide information about the rate of change of a response variable. They can also indicate the interactions among the quantitative treatment factors. The special class of designed experiments for fitting the response surfaces is called response surface design. Response surface designs have their wide applications in agricultural and industrial experiments. Similar to factorial experiments the response surface design may exhibit trend over space or time. Hinkelmann and Jo (1998) first studied linear trend-free response surface designs. They gave a procedure to construct the linear trend-free Box-Behnken response surface designs by using the solutions of some of the equations.

4.2 Box-Behnken designs

A class of three-level incomplete factorial designs for the estimation of the parameters in second order model was developed by Box and Behnken (1960). By definition, a three-level incomplete factorial design is a subset of the factorial combinations from a 3^k factorial design. The Box-Behnken designs are formed by combining two-level factorial designs with balanced incomplete block (BIB) designs or partially balanced incomplete block (PBIB) designs in a particular manner.

Consider a binary incomplete block design with v treatments, b blocks, r replications and k units per block. Let $\mathbf{N} = (n_{ij})$ (i = 1, 2, …, v ; j = 1, 2, ..., b) denotes the incidence matrix of the BIB or PBIB design. Then we prepare a 2^k factorial (where k is the same as the block size for the given incomplete block design). The levels of k factors are denoted by -1 and +1, representing the low and high level respectively. We write the factorial experiment in standard (lexicographic) order. We then write k column vectors $\mathbf{A}_i$ (i = 1, 2, …, k) of length 2^k with elements ±1 such that the elements of $\mathbf{A}_i$ add to zero and $\mathbf{A}_i$'s are orthogonal to each other.

Box and Behnken (1960) took the $\mathbf{A}_i$ to be the columns of the 2^k factorial treatment combinations; other choices are also possible, such as the coefficient vector for the main effects or interactions for the 2^k factorial. Finally, we substitute in each row of $\mathbf{N}'$ the first 1 by $\mathbf{A}_1$, the second 1 by $\mathbf{A}_2$ and so on the k^{th} 1 by $\mathbf{A}_k$ and all 0's by $\mathbf{0}$ column vectors of size 2^k. Same procedure will be repeated in all rows of the transpose of incidence matrix *i.e.* $\mathbf{N}'$. So there are total $b2^k$ rows, represents a run for the v input variables each taking the values -1, 0, +1. To this obtained design we add n_0 central runs (0, 0, …, 0) to obtain a final design, n_0 should be odd in numbers. This is Box-Behnken design with total number of runs $n = b\,2^k + n_0$ runs in general.

4.3 Linear trend-free Box- Behnken design

The treatments combinations obtained in 4.2 are applied randomly to the *n* experimental units. If it is, however, known or assumed that the experimental units or the experimental material applied to the experimental units exhibit a linear trend over time or space, then it will be more advantageous to choose a systematic arrangement of the *n* run, such that the resulting design is a linear trend-free design.

For the second order response surface design we consider the model

$$\mathbf{y} = \beta_0\mathbf{1} + \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \mathbf{X}_3\boldsymbol{\beta}_3 + \mathbf{T}\boldsymbol{\theta} + \mathbf{e} \tag{4.1}$$

where,

$\mathbf{y}$ is $n \times 1$ vector of observations,

$\mathbf{X}_1$ is $n \times v$ matrix of linear effects,

$\mathbf{X}_2$ is $n \times v$ matrix of quadratic effects,

$\mathbf{X}_3$ is $n \times v(v-1)/2$ matrix of cross products,

β_0 is the vector of a constant,

$\boldsymbol{\beta}_1$ is the vector of linear coefficients,

$\boldsymbol{\beta}_2$ is the vector of quadratic coefficients,

$\boldsymbol{\beta}_3$ is the vector of cross product second order coefficient,

T is the vector of coefficients of the first-degree orthogonal polynomial of order *n*,

θ is a regression coefficient, and **e** is a $n \times 1$ vector of independently and identically (normally) distributed errors.

Based on a more general definition of linear trend-free, the design under model (4.1) is linear trend free if the conditions $\mathbf{X}_\mathbf{i}'\mathbf{T} = \mathbf{0}$ (for i = 1, 2, 3) are satisfied.

In this objective we develop the algorithm of linear trend free Box-Behnken response surface designs.

We have a Box-Behnken design with $n = b\,2^k + n_0$ runs, n_0 is odd in numbers. We can write vector **T** for odd number of runs as

$$\mathbf{T}' = \left(-\frac{n-1}{2}, -\frac{n-1}{2} + 1, \ldots, -2, -1, 0, 1, 2, \ldots, \frac{n-1}{2} - 1, \frac{n-1}{2} \right)$$

(4.2)

T is antisymmetric.

A general procedure for the development of algorithm of linear trend-free Box-Behnmen response surface design is obtained for $k \geq 4$. For k = 2 no general procedure is obtained and design is obtained by solving the equations. No algorithm is given for k = 2. The design is obtained manually by solving the equations and is included in the catalogue. So no algorithm is given for k = 2, although the design is included in the catalogue. The algorithm for generation/search of linear trend-free Box-Behnken response surface design for $k \geq 4$ is given below.

4.4 Algorithm for the linear trend-free Box- Behnken design for *k* ≥ 4

I.1 Take a binary incomplete block (BIB) design with v treatments, b blocks, r replications and k units per block. This is stored in a $b \times k$ array.

I.2 Prepare a 2^k factorial (where k is the same as the block size for the BIB design) in standard order as described in step I.1 to II.8 under Section 2.3.

I.3 The levels of k factors are denoted by -1 and +1, representing the low and high level respectively

I.4 From the above mentioned binary incomplete block (BIB) design in $b \times k$ array, prepare incidence matrix in $v \times b$ array, by reading the values block wise and putting the value 1 in the columns which are the treatment number. The obtained matrix is incidence matrix **N** of binary incomplete block (BIB) design.

Let **N** = (n_{ij}) (i = 1, 2, . . ., v ; j = 1, 2, . . ., b).

I.5 Take transpose of the incidence matrix **N,** *i.e.* $\mathbf{N}'$. The matrix $\mathbf{N}'$ has as many ones in each row as number of columns in 2^k factorial experiment.

I.6 Let A_i, i = 1, 2, …, k represent the coefficient of the contrasts of the k main effects. Perform component-wise product within each of b blocks separately. Similar to complete factorial experiment, there are two procedures of component-wise product. The procedure of component-wise product is as given below:

For *k* is odd: When k is odd the new coefficient vectors for main effects will be obtained as:

Let A_i be the new coefficient vectors for main effects for ith factor, $i = 1, 2, \ldots, k$

$$F_i = A_1 \circ A_2 \circ \cdots \circ A_{i-1} \circ A_{i+1} \circ \cdots \circ A_k \,;\; (i = 1, 2, \ldots, k-1)$$

and $F_k = A_1 \circ A_2 \circ \cdots \circ A_k$; $(i = k)$

For *k* is even: When k is even the new coefficient vectors for main effects will be obtained as:

$$F_i = A_1 \circ A_2 \circ \cdots \circ A_{i-1} \circ A_{i+1} \circ \cdots \circ A_k \,;\; (i = 1, 2, \ldots, k).$$

I.7 Replace each 1 in matrix $\mathbf{N}'$ by obtained F_i, such that first 1 by first column (F_1), second 1 by second column (F_2) and so on, 0's by a columns of zero.

I.8 This choice of F_i ensures that they are independent in the sense that no F_i is the generalized interaction of F_j's. Also, since $k \geq 4$ it follows that each F_i corresponds to at least a three factor interaction.

I.9 The treatment combinations of the 2^k factorial in standard order and the result of Cheng and Jacroux (1988) implies that each half is F_i, F_i^U and F_i^L for $(i = 1, 2, \ldots, k)$ is orthogonal to a linear trend. Say this design is **D***

I.10 We can write **D*** as, $\mathbf{D^*} = (\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_b})'$, Each B_i $(i = 1, 2, \ldots, b)$ contains $F_1, F_2, \ldots, F_K$ exactly once, and v – k **0**'s

Partition of each B_i is done into an upper part and a lower part as:

$$\mathbf{B_i} = \begin{bmatrix} \mathbf{B_i^U} \\ \mathbf{B_i^L} \end{bmatrix} (i = 1, 2, \ldots, b)$$

Arrange all $\mathbf{B}_i^U$ in $\mathbf{P}_1$ and $\mathbf{B}_i^L$ in $\mathbf{P}_2$ and $\mathbf{P}_0$ is the central runs (in odd number) partitioning the design matrix **D***, we obtain a design matrix **D** as

$$\mathbf{D} = \begin{bmatrix} \mathbf{P_1} \\ \mathbf{P_0} \\ \mathbf{P_2} \end{bmatrix}$$

where $\mathbf{P}_1$ represents the first $b2^{k-1}$ runs, $\mathbf{P}_2$ represents the last $b2^{k-1}$ runs and $\mathbf{P}_0$ is the central runs (in odd number).

I.11 The resultant design is trend free Box-Behnken Design. Theses designs are linear trend-free for all effects: (a) linear effects are linear trend-free as shown in Section 2.4, hence, $\mathbf{X}_1'\mathbf{T} = \mathbf{0}$. (b) F_{ii} and $F_{ii'}$ are symmetric, so $\mathbf{X}_2'\mathbf{T} = \mathbf{0}$ and $\mathbf{X}_3'\mathbf{T} = \mathbf{0}$ implies that quadratic and cross product effects are also linear trend-free.

For k = 3, no procedure could be worked out by solving equations. However, we followed the same procedure as given above (for $k \geq 4$). From the procedure, Box-Behnken Design is obtained. In this design, all four the linear and four quadratic effects are linear trend-free. Among six cross products effects, one effect is linear trend-free, two are nearly linear trend-free and three are neither nearly nor linear trend-free. This design is also included in the catalogue.

4.5 Working of algorithm for $k \geq 4$

Let we have a BIB design with parameters (v = 5, b = 5, r = 4, k = 4, λ = 3).

$$\mathbf{N} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 5 \\ 1 & 2 & 4 & 5 \\ 1 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 \end{bmatrix}$$

Generate incidence matrix from step I.4

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Generate its transpose matrix

$$\mathbf{N}' = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Since block size is 4 hence 2^4 factorial design in lexicographic order is generated. Using steps I.1 to II.8 under Section 2.3

Design – 4.1

Treatment combinations	A_a	A_b	A_c	A_d
(1)	-1	-1	-1	-1
a	1	-1	-1	-1
b	-1	1	-1	-1
ab	1	1	-1	-1
c	-1	-1	1	-1
ac	1	-1	1	-1
bc	-1	1	1	-1
abc	1	1	1	-1
d	-1	-1	-1	1
ad	1	-1	-1	1
bd	-1	1	-1	1
abd	1	1	-1	1
cd	-1	-1	1	1
acd	1	-1	1	1
bcd	-1	1	1	1
abcd	1	1	1	1

Using computer program of Algorithm-II of chapter-II, perform the component-wise product of A_i and A_j by using step II.4. Here *k* (= 4) is even so factor A has been generated using component-wise product of A_b, A_c and A_d as: $A_a = A_b \circ A_c \circ A_d$.

A_b	A_c	A_d	$A_b \circ A_c \circ A_d$		A
-1	-1	-1	-1*-1*-1	=	-1
-1	-1	-1	-1*-1*-1	=	-1
1	-1	-1	1*-1*-1	=	1
1	-1	-1	1*-1*-1	=	1
-1	1	-1	-1*1*-1	=	1
-1	1	-1	-1*1*-1	=	1
1	1	-1	1*1*-1	=	-1
1	1	-1	1*1*-1	=	-1
-1	-1	1	-1*-1*1	=	1
-1	-1	1	-1*-1*1	=	1
1	-1	1	1*-1*1	=	-1
1	-1	1	1*-1*1	=	-1
-1	1	1	-1*1*1	=	-1
-1	1	1	-1*1*1	=	-1
1	1	1	1*1*1	=	1
1	1	1	1*1*1	=	1

Further B, C and D are generated using the formula:

$F_i = A_1 \circ A_2 \circ \cdots \circ A_{i-1} \circ A_{i+1} \circ \cdots \circ A_k$; $(i = a, b, c,$ and $d)$

Thus we get the Design-4.2

Design-4.2

Treatment combinations	**A (**F_a**)**	**B (**F_b**)**	**C (**F_c**)**	**D (**F_d**)**
(1)	-1	-1	-1	-1
bcd	-1	1	1	1
acd	1	-1	1	1
ab	1	1	-1	-1
abd	1	1	-1	1
ac	1	-1	1	-1
bc	-1	1	1	-1
d	-1	-1	-1	1
abc	1	1	1	-1
ad	1	-1	-1	1
bd	-1	1	-1	1
c	-1	-1	1	-1

cd	-1	-1	1	1
b	-1	1	-1	-1
a	1	-1	-1	-1
abcd	1	1	1	1

Substituting each column of factorial experiment in transpose incidence matrix, we get the Box Behnken design in five factors each at three levels in 80 runs.

A	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
-1	-1	-1	-1	0	1	1	0	1	-1
-1	1	1	1	0	1	-1	0	-1	1
1	-1	1	1	0	-1	1	0	-1	1
1	1	-1	-1	0	-1	-1	0	1	-1
1	1	-1	1	0	-1	-1	0	1	1
1	-1	1	-1	0	-1	1	0	-1	-1
-1	1	1	-1	0	1	-1	0	-1	-1
-1	-1	-1	1	0	1	1	0	1	1
1	1	1	-1	0	-1	0	-1	-1	-1
1	-1	-1	1	0	-1	0	1	1	1
-1	1	-1	1	0	1	0	-1	1	1
-1	-1	1	-1	0	1	0	1	-1	-1
-1	-1	1	1	0	1	0	1	-1	1
-1	1	-1	-1	0	1	0	-1	1	-1
1	-1	-1	-1	0	-1	0	1	1	-1
1	1	1	1	0	-1	0	-1	-1	1
-1	-1	-1	0	-1	1	0	1	1	-1
-1	1	1	0	1	1	0	-1	-1	1
1	-1	1	0	1	-1	0	1	-1	1
1	1	-1	0	-1	-1	0	-1	1	-1
1	1	-1	0	1	-1	0	-1	1	1
1	-1	1	0	-1	-1	0	1	-1	-1
-1	1	1	0	-1	1	0	-1	-1	-1
-1	-1	-1	0	1	1	0	1	1	1
1	1	1	0	-1	0	-1	-1	-1	-1
1	-1	-1	0	1	0	-1	1	1	1
-1	1	-1	0	1	0	1	-1	1	1
-1	-1	1	0	-1	0	1	1	-1	-1
-1	-1	1	0	1	0	1	1	-1	1
-1	1	-1	0	-1	0	1	-1	1	-1
1	-1	-1	0	-1	0	-1	1	1	-1
1	1	1	0	1	0	-1	-1	-1	1
-1	-1	0	-1	-1	0	1	1	1	-1
-1	1	0	1	1	0	1	-1	-1	1
1	-1	0	1	1	0	-1	1	-1	1
1	1	0	-1	-1	0	-1	-1	1	-1

1	1	0	-1	1	0	-1	-1	1	1
1	-1	0	1	-1	0	-1	1	-1	-1
-1	1	0	1	-1	0	1	-1	-1	-1
-1	-1	0	-1	1	0	1	1	1	1

We can write **D*** as: $\mathbf{D^*} = (\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_4})'$

Each $B_i\ (i = a, b, c, d, e)$ contains $F_a, F_b \ldots, F_d$ exactly once, and 1 column of zero.

Now partition of each $\mathrm{B_i}$ is done into an upper part and a lower part as: $\mathbf{B_i} = \begin{bmatrix} \mathbf{B_i^U} \\ \mathbf{B_i^L} \end{bmatrix} (i = 1, 2, \ldots, b)$

Rearranging and partitioning the design matrix **D***, we obtain a design matrix **D** as

$$\mathbf{D} = \begin{bmatrix} \mathbf{P_1} \\ \mathbf{P_0} \\ \mathbf{P_2} \end{bmatrix}$$

where $\mathbf{P}_1$ represents the first 40 runs, $\mathbf{P}_2$ represents the last 40 runs and $\mathbf{P}_0$ has one central runs. Thus the obtained Box-Behnken design in 80 runs is linear trend-free for all five linear effects, five quadratic effects and ten cross products.

A	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
-1	-1	-1	-1	0	0	1	1	1	-1
-1	1	1	1	0	0	1	-1	-1	1
1	-1	1	1	0	0	-1	1	-1	1
1	1	-1	-1	0	0	-1	-1	1	-1
1	1	-1	1	0	0	-1	-1	1	1
1	-1	1	-1	0	0	-1	1	-1	-1
-1	1	1	-1	0	0	1	-1	-1	-1
-1	-1	-1	1	0	0	1	1	1	1
-1	-1	-1	0	-1	1	0	1	1	-1
-1	1	1	0	1	1	0	-1	-1	1
1	-1	1	0	1	-1	0	1	-1	1
1	1	-1	0	-1	-1	0	-1	1	-1
1	1	-1	0	1	-1	0	-1	1	1
1	-1	1	0	-1	-1	0	1	-1	-1
-1	1	1	0	-1	1	0	-1	-1	-1
-1	-1	-1	0	1	1	0	1	1	1
-1	-1	0	-1	-1	1	1	0	1	-1
-1	1	0	1	1	1	-1	0	-1	1
1	-1	0	1	1	-1	1	0	-1	1
1	1	0	-1	-1	-1	-1	0	1	-1
1	1	0	-1	1	-1	-1	0	1	1

1	-1	0	1	-1	-1	1	0	-1	-1
-1	1	0	1	-1	1	-1	0	-1	-1
-1	-1	0	-1	1	1	1	0	1	1
-1	0	-1	-1	-1	1	1	1	0	-1
-1	0	1	1	1	1	-1	-1	0	1
1	0	-1	1	1	-1	1	-1	0	1
1	0	1	-1	-1	-1	-1	1	0	-1
1	0	1	-1	1	-1	-1	1	0	1
1	0	-1	1	-1	-1	1	-1	0	-1
-1	0	1	1	-1	1	-1	-1	0	-1
-1	0	-1	-1	1	1	1	1	0	1
0	-1	-1	-1	-1	1	1	1	-1	0
0	-1	1	1	1	1	-1	-1	1	0
0	1	-1	1	1	-1	1	-1	1	0
0	1	1	-1	-1	-1	-1	1	-1	0
0	1	1	-1	1	-1	-1	1	1	0
0	1	-1	1	-1	-1	1	-1	-1	0
0	-1	1	1	-1	1	-1	-1	-1	0
0	-1	-1	-1	1	1	1	1	1	0
0	0	0	0	0					

Appendix-4.1

Computer program in Microsoft Visual C++ for obtaining linear trend-free Box-Behnken designs

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
static int a[1024][10],b[10][6],b1[1024][10],c[10][10],arr[80][10],k;
void main()
{
        FILE *fp;
        fp=fopen("f:\sush1.txt","a+");
        int i,j,h=0,l,m,m1,m2,v,n,y=0,z=0,e;
printf("Enter the no of treatments v= ");
scanf("%d",&v);
printf("Enter the block size K = ");
scanf("%d",&n);
printf("Enter the No of blocks b = ");
scanf("%d",&e);
printf("Enter the treatment No row wise ");
for(i=0;i<e;i++)
{
        for(j=0;j<n;j++)
        {
                scanf("%d",&b[i][j]);
        }
}
printf("Enter the number of central points");
scanf("%d",&m2);
if(fp!=NULL)
{
m=pow(2,n);
for(i=0;i<n;i++)
        y=y+i;
for(j=0;j<n;j++)
{
  if(j==0)
   {
   for(i=0;i<m;i++)
    {
     if(i%2==0)
     a[i][j]=-1;
     else
     a[i][j]=1;
    }
```

```
    }
   else
   {
     for(i=0;i<m;)
     {
          if(h%2==0)
          {
            for(l=0;l<pow(2,j);l++)
             {
              a[i][j]=-1;
              i++;
             }
          }
          else
          {
            for(l=0;l<pow(2,j);l++)
             {
              a[i][j]=1;
              i++;
             }
          }
    h++;
    }
}
h=0;
}
for(i=0;i<e;i++)
    {
    for(j=0;j<n;j++)
                        {
                        fprintf(fp,"%3d",b[i][j]);
                                    fprintf(fp,"\t");
                        }
                                                            fprintf(fp,"\n");
          }
if(n>=4)
{
if (n%2==0)
{
     for (i=0;i<m;i++)
      {
          for (k=0;k<n;k++)
           {
            b1[i][k]=1;
              for(j=0;j<n;j++)
                b1[i][k]= b1[i][k]*a[i][j];
```

```
                b1[i][k]=b1[i][k]/a[i][k];
              }
}
}
else
{
     for (i=0;i<m;i++)
      {
            for (k=0;k<n-1;k++)
             {
              b1[i][k]=1;
                for(j=0;j<n;j++)
                 b1[i][k]= b1[i][k]*a[i][j];
                 b1[i][k]=b1[i][k]/a[i][k];
              }
            {


                 b1[i][k]=1;
               for(j=0;j<n;j++)
                 b1[i][k]= b1[i][k]*a[i][j];

              }
}
}
for(i=0;i<m;i++)
    {
            for(k=0;k<n;k++)
            {
            fprintf(fp,"%3d",b1[i][k]);
            fprintf(fp,"\t");
            }
            fprintf(fp,"\n");
    }
            for(i=0;i<v;i++)
            {
                        for(j=0;j<e;j++)
                                    c[i][j]=0;
            }
for(i=0;i<e;i++)
    {
    for(j=0;j<n;j++)
                        {
                                    m1=0;
                                    m1=b[i][j];
                                    c[m1-1][i]=1;
                        }
```

```
}
for(i=0;i<v;i++)
          {
                    for(j=0;j<e;j++)
                    {
                              fprintf(fp,"%3d",c[i][j]);
                              fprintf(fp,"\t");
                    }
                    fprintf(fp,"\n");
}

int trans_mat[10][10];
fprintf(fp,"this is the transpose matrix\n");
for (i=0;i<e;i++)
{
          for (j=0;j<v;j++)
          {
                    trans_mat[i][j]=c[j][i];
                    fprintf(fp,"%3d",trans_mat[i][j]);
                    fprintf(fp,"\t");
          }

          fprintf(fp,"\n");
}
int c1[100][10];
for(i=0;i<e*m+m2;i++)
          {
                              for(j=0;j<v;j++)
                              c1[i][j]=0;
          }
int i1,i2=0;
for(i=0;i<e;i++)
{
          i2=0;
          for(j=0;j<v;j++)
          {
                    if(trans_mat[i][j]==1)
                    {
                              for(i1=i*m;i1<(i+1)*m;i1++)
                              c1[i1][j]=b1[i1%m][i2];
                              i2++;
                    }

          }

}
```

```
int d[1024][10];
for(i=0;i<e;i++)
{
        i2=0,i1=0;
        for(j=0;j<v;j++)
        {
                if(trans_mat[i][j]==1)
                {
                        for(i1=i*m;i1<(i+1)*m;i1++)
                        d[i1][j]=b1[i1%m][i2];
                        i2++;
                }
        }
}
for(i=0;i<e*m+m2;i++)
        {
                for(j=0;j<v;j++)
                {
                        fprintf(fp,"%3d",c1[i][j]);
                        fprintf(fp,"\t");
                }
                fprintf(fp,"\n");
        }

int c2[20][5],k=0;
for(i=0;i<e*m;i++)
        {
                if((i%m)<(m/2))
                {
                        for(j=0;j<v;j++)
                        {
                                c2[k][j]=c1[i][j];
                                fprintf(fp,"%3d",c2[k][j]);
                                fprintf(fp,"\t");
                        }
                        k++;fprintf(fp,"\n");
                }

        }
}
if(n<4)
{
for(i=0;i<m;i++)
   {
        //fprintf(fp,"block[%d] = ",i+1);
   for(j=0;j<n;j++)
```

```
                {
                fprintf(fp,"%3d",a[i][j]);
                        fprintf(fp,"\t");
                }

                                        fprintf(fp,"\n");
        }
        for(i=0;i<v;i++)
        {
                for(j=0;j<e;j++)
                        c[i][j]=0;
        }
for(i=0;i<e;i++)
   {
   for(j=0;j<n;j++)
                {
                        m1=0;
                        m1=b[i][j];
                        c[m1-1][i]=1;
                }
}
for(i=0;i<v;i++)
        {
                for(j=0;j<e;j++)
                {
                        fprintf(fp,"%3d",c[i][j]);
                        fprintf(fp,"\t");
                }
                fprintf(fp,"\n");
}

int trans_mat[10][10];
fprintf(fp,"this is the transpose matrix\n");
for (i=0;i<e;i++)
{
        for (j=0;j<v;j++)
        {
                trans_mat[i][j]=c[j][i];
                fprintf(fp,"%3d",trans_mat[i][j]);
                fprintf(fp,"\t");
        }

        fprintf(fp,"\n");
}

int c1[100][10];
```

```
for(i=0;i<e*m+m2;i++)
          {
                              for(j=0;j<v;j++)
                              c1[i][j]=0;
          }
int i1,i2=0;
for(i=0;i<e;i++)
{
          i2=0;
          for(j=0;j<v;j++)
          {
                    if(trans_mat[i][j]==1)
                    {
                              for(i1=i*m;i1<(i+1)*m;i1++)
                              c1[i1][j]=a[i1%m][i2];
                              i2++;
                    }

          }

}
int d[1024][10];
for(i=0;i<e;i++)
{
          i2=0,i1=0;
          for(j=0;j<v;j++)
          {
                    if(trans_mat[i][j]==1)
                    {
                              for(i1=i*m;i1<(i+1)*m;i1++)
                              d[i1][j]=a[i1%m][i2];
                              i2++;
                    }
          }
}
for(i=0;i<e*m+m2;i++)
          {
                    for(j=0;j<v;j++)
                    {
                              fprintf(fp,"%3d",c1[i][j]);
                              fprintf(fp,"\t");
                    }
                    fprintf(fp,"\n");
          }
int c2[20][5],k=0;
for(i=0;i<e*m;i++)
```

```
		{
			if((i%m)<(m/2))
			{
				for(j=0;j<v;j++)
				{
					c2[k][j]=c1[i][j];
					fprintf(fp,"%3d",c2[k][j]);
					fprintf(fp,"\t");
				}
				k++;fprintf(fp,"\n");
			}

		}
	}
	}
	fclose(fp);
	}
```

Appendix-4.2

Catalogue of linear trend-free Box-Behnken designs (for *k* = 2, 3, 4 and 5)

Design-1: for *k* = 2, Box-Behnken design in 3 factors

A	B	C
1	1	0
-1	0	-1
-1	0	1
1	-1	0
0	-1	1
0	-1	-1
0	0	0
0	1	1
0	1	-1
-1	1	0
1	0	-1
1	0	1
-1	-1	0

All effects are linear trend-free

Design-2: for *k* = 3, Box-Behnken design in 4 factors

A	B	C	D	A	B	C	D
1	1	-1	0	1	-1	1	0
-1	-1	1	0	-1	1	-1	0
-1	1	1	0	-1	-1	-1	0
1	-1	-1	0	1	1	1	0
1	1	0	-1	1	-1	0	1
-1	-1	0	1	-1	1	0	-1
-1	1	0	1	-1	-1	0	-1
1	-1	0	-1	1	1	0	1
1	0	1	-1	1	0	-1	1
-1	0	-1	1	-1	0	1	-1
-1	0	1	1	-1	0	-1	-1
1	0	-1	-1	1	0	1	1
0	1	1	-1	0	1	-1	1
0	-1	-1	1	0	-1	1	-1
0	-1	1	1	0	-1	-1	-1
0	1	-1	-1	0	1	1	1
0	0	0	0				

All linear, quadratic effects and AB are linear trend-free, BC, CD is nearly linear trend-free and others are neither linear nor nearly linear trend-free.

Design-3: for $k = 4$, Box-Behnken design in 5 factors

A	B	C	D	E
-1	-1	-1	-1	0
-1	1	1	1	0
1	-1	1	1	0
1	1	-1	-1	0
1	1	-1	1	0
1	-1	1	-1	0
-1	1	1	-1	0
-1	-1	-1	1	0
-1	-1	-1	0	-1
-1	1	1	0	1
1	-1	1	0	1
1	1	-1	0	-1
1	1	-1	0	1
1	-1	1	0	-1
-1	1	1	0	-1
-1	-1	-1	0	1
-1	-1	0	-1	-1
-1	1	0	1	1
1	-1	0	1	1
1	1	0	-1	-1
1	1	0	-1	1
1	-1	0	1	-1
-1	1	0	1	-1
-1	-1	0	-1	1
-1	0	-1	-1	-1
-1	0	1	1	1
1	0	-1	1	1
1	0	1	-1	-1
1	0	1	-1	1
1	0	-1	1	-1
-1	0	1	1	-1
-1	0	-1	-1	1
0	-1	-1	-1	-1
0	-1	1	1	1
0	1	-1	1	1
0	1	1	-1	-1
0	1	1	-1	1
0	1	-1	1	-1
0	-1	1	1	-1
0	-1	-1	-1	1
0	0	0	0	0
0	1	1	1	-1
0	1	-1	-1	1
0	-1	1	-1	1
0	-1	-1	1	-1
0	-1	-1	1	1
0	-1	1	-1	-1
0	1	-1	-1	-1
0	1	1	1	1
1	0	1	1	-1
1	0	-1	-1	1
-1	0	1	-1	1
-1	0	-1	1	-1
-1	0	-1	1	1
-1	0	1	-1	-1
1	0	-1	-1	-1
1	0	1	1	1
1	1	0	1	-1
1	-1	0	-1	1
-1	1	0	-1	1
-1	-1	0	1	-1
-1	-1	0	1	1
-1	1	0	-1	-1
1	-1	0	-1	-1
1	1	0	1	1
1	1	1	0	-1
1	-1	-1	0	1
-1	1	-1	0	1
-1	-1	1	0	-1
-1	-1	1	0	1
-1	1	-1	0	-1
1	-1	-1	0	-1
1	1	1	0	1
1	1	1	-1	0
1	-1	-1	1	0
-1	1	-1	1	0
-1	-1	1	-1	0
-1	-1	1	1	0
-1	1	-1	-1	0
1	-1	-1	-1	0
1	1	1	1	0

All effects are linear trend-free

Design-4: for *k* = 5, Box-Behnken design in 6 factors

A	B	C	D	E	F
1	1	1	1	0	-1
1	-1	-1	-1	0	1
-1	1	-1	-1	0	1
-1	-1	1	1	0	-1
-1	-1	1	-1	0	1
-1	1	-1	1	0	-1
1	-1	-1	1	0	-1
1	1	1	-1	0	1
-1	-1	-1	1	0	1
-1	1	1	-1	0	-1
1	-1	1	-1	0	-1
1	1	-1	1	0	1
1	1	-1	-1	0	-1
1	-1	1	1	0	1
-1	1	1	1	0	1
-1	-1	-1	-1	0	-1
1	1	1	1	-1	0
1	-1	-1	-1	1	0
-1	1	-1	-1	1	0
-1	-1	1	1	-1	0
-1	-1	1	-1	1	0
-1	1	-1	1	-1	0
1	-1	-1	1	-1	0
1	1	1	-1	1	0
-1	-1	-1	1	1	0
-1	1	1	-1	-1	0
1	-1	1	-1	-1	0
1	1	-1	1	1	0
1	1	-1	-1	-1	0
1	-1	1	1	1	0
-1	1	1	1	1	0
-1	-1	-1	-1	-1	0
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
1	0	1	1	1	-1
1	0	-1	-1	-1	1
-1	0	1	-1	-1	1
-1	-1	-1	0	-1	1
-1	1	1	0	1	-1
1	-1	1	0	1	-1
1	1	-1	0	-1	1
1	1	-1	0	1	-1
1	-1	1	0	-1	1
-1	1	1	0	-1	1
-1	-1	-1	0	1	-1
1	1	1	0	-1	-1
1	-1	-1	0	1	1
-1	1	-1	0	1	1
-1	-1	1	0	-1	-1
-1	-1	1	0	1	1
-1	1	-1	0	-1	-1
1	-1	-1	0	-1	-1
1	1	1	0	1	1
-1	-1	0	-1	-1	1
-1	1	0	1	1	-1
1	-1	0	1	1	-1
1	1	0	-1	-1	1
1	1	0	-1	1	-1
1	-1	0	1	-1	1
-1	1	0	1	-1	1
-1	-1	0	-1	1	-1
1	1	0	1	-1	-1
1	-1	0	-1	1	1
-1	1	0	-1	1	1
-1	-1	0	1	-1	-1
-1	-1	0	1	1	1
-1	1	0	-1	-1	-1
1	-1	0	-1	-1	-1
1	1	0	1	1	1
-1	0	-1	-1	-1	1
-1	0	1	1	1	-1
1	0	-1	1	1	-1
1	0	1	-1	-1	1
1	0	1	-1	1	-1
1	0	-1	1	-1	1
-1	0	1	1	-1	1
-1	0	-1	-1	1	-1
1	0	1	1	-1	-1
1	0	-1	-1	1	1
-1	0	1	-1	1	1
-1	0	-1	1	-1	-1
-1	0	-1	1	1	1
-1	0	1	-1	-1	-1
1	0	-1	-1	-1	-1
1	0	1	1	1	1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1

-1	0	-1	1	1	-1
-1	0	-1	1	-1	1
-1	0	1	-1	1	-1
1	0	-1	-1	1	-1
1	0	1	1	-1	1
-1	0	-1	-1	1	1
-1	0	1	1	-1	-1
1	0	-1	1	-1	-1
1	0	1	-1	1	1
1	0	1	-1	-1	-1
1	0	-1	1	1	1
-1	0	1	1	1	1
-1	0	-1	-1	-1	-1
1	1	0	1	1	-1
1	-1	0	-1	-1	1
-1	1	0	-1	-1	1
-1	-1	0	1	1	-1
-1	-1	0	1	-1	1
-1	1	0	-1	1	-1
1	-1	0	-1	1	-1
1	1	0	1	-1	1
-1	-1	0	-1	1	1
-1	1	0	1	-1	-1
1	-1	0	1	-1	-1
1	1	0	-1	1	1
1	1	0	-1	-1	-1
1	-1	0	1	1	1
-1	1	0	1	1	1
-1	-1	0	-1	-1	-1
1	1	1	0	1	-1
1	-1	-1	0	-1	1
-1	1	-1	0	-1	1
-1	-1	1	0	1	-1
-1	-1	1	0	-1	1
-1	1	-1	0	1	-1
1	-1	-1	0	1	-1
1	1	1	0	-1	1
-1	-1	-1	0	1	1
-1	1	1	0	-1	-1
1	-1	1	0	-1	-1
1	1	-1	0	1	1
1	1	-1	0	-1	-1
1	-1	1	0	1	1
-1	1	1	0	1	1
-1	-1	-1	0	-1	-1
0	0	0	0	0	0

0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
0	1	1	1	1	-1
-1	-1	-1	-1	1	0
-1	1	1	1	-1	0
1	-1	1	1	-1	0
1	1	-1	-1	1	0
1	1	-1	1	-1	0
1	-1	1	-1	1	0
-1	1	1	-1	1	0
-1	-1	-1	1	-1	0
1	1	1	-1	-1	0
1	-1	-1	1	1	0
-1	1	-1	1	1	0
-1	-1	1	-1	-1	0
-1	-1	1	1	1	0
-1	1	-1	-1	-1	0
1	-1	-1	-1	-1	0
1	1	1	1	1	0
-1	-1	-1	-1	0	1
-1	1	1	1	0	-1
1	-1	1	1	0	-1
1	1	-1	-1	0	1
1	1	-1	1	0	-1
1	-1	1	-1	0	1
-1	1	1	-1	0	1
-1	-1	-1	1	0	-1
1	1	1	-1	0	-1
1	-1	-1	1	0	1
-1	1	-1	1	0	1
-1	-1	1	-1	0	-1
-1	-1	1	1	0	1
-1	1	-1	-1	0	-1
1	-1	-1	-1	0	-1
1	1	1	1	0	1

All effects are linear trend-free.

SUMMARY

Designs for factorial experiments have been widely used in agricultural, biological and industrial experiments. The experimental units in a design of a factorial experiment may exhibit a trend over space or time. Such situations may occur in agricultural experiments when there is a slope in the field and there is sequential application of the treatments to the same experimental unit. This may also happen when the land is irrigated, the nutrients supplied by the fertilizers may not be equally distributed and trend in experimental units may be due to slope. Other experimental situations where trend may occur are Green-house experiments, in which source of heat is located on sides of the house, in poultry experiments where the source of heat is at the centre of the shed and chicks of early age are in the cages, in animal experiments where littermates (animals born in the same litter) are experimental units within a block *i.e.* litters are blocks, orchard and vineyard experiments on undulating topography experiments in which response variable of interest is affected by slowly migrating insects entering the area from one side, laboratory experiments where the responses to the experimental units may be affected within time periods by instrument drift or analyst fatigue, etc.

In factorial experiments, interest is in estimation of main effects, interactions of two factors, three factors and so on and testing the hypothesis with respect to these effects. If there is trend in the experimental units, the interest of experimenter is to eliminate the trend effect and obtain the estimates free from trend effects. Mostly the trend is linear in nature. Thus in the presence of linear trend in the experimental units in factorial experiments, the treatments combinations of factorial experiment are allocated to the experimental units such that the contrasts of interest (main effects and lower order interactions) are estimated free from linear-trend. The resulting designs are called as *linear trend-free designs for factorial experiments* for the effects of interest and ordered application of treatments to experimental units is called *run order*.

In block designs for single factor experiments sufficient literature is available on trend-free designs but on factorial experiments. But sufficient literature on trend-free factorial

experiments particularly on trend-free confounded factorial experiments, trend-free fractional factorial experiments and trend-free response surface designs could not be traced. In factorial experiments, the experimenter is always interested to obtain trend-free designs for various treatment effects. Obtaining trend free designs for multifactor experiments are tedious as well as a lot of algebra is involved. As a consequence linear trend free designs for factorial experiments, whatever little available have not found much favour from the experimenters.

One of the possible reasons of hampering the applications of these designs is that the construction for such design is not easily available. Therefore, it is required to provide easy method of construction, possibly computer aided generation of designs. Complete, confounded, fractional designs that are at least linear trend-free for main effects. Further it is required to search/ identify low order interaction effects, that are estimable linear or nearly linear trend-free from these designs. In many factorial experiments, the factors are quantitative in nature and it is desired to establish a relationship between response and level of various factors. For such situations, response surface designs are used. The run orders in response surface design may also be affected by the presence of trend. Thus, there is a need to make attempt for computer aided construction of linear trend-free response surface designs. This research work is devoted to the development of computer algorithms for construction of trend-free designs for multifactor experimental settings. The algorithm developed is helpful in generating the factorial experiments that are linear trend-free for main effects. A search has been made to identify the two and three factor interactions that are estimable free from linear trend effect. The developed algorithm is in general and can generate linear trend-free designs for any number of factors each at two levels. The algorithm has also been developed for generation of two level linear trend-free fractional factorial plans. Algorithm has also been developed to generate Box-Behnken response surface designs that are linear trend-free in general and a catalogue for linear trend-free Box-Behnken response surface designs. This thesis has been divided into four Chapters.

Some of the experimental situations that may have a trend over experimental units have been elaborated in Chapter-I. The meaning of factorial experiments that are linear trend-free for its main effects and other treatment combinations which are of the interest to the experimenter is discussed. It is followed by a review of literature, motivation of the problem and objectives of the studies.

Development of algorithm to generate/ construct complete factorial experiments that are linear trend-free for main effects, using the technique of component-wise product, is given in Chapter-II. The algorithm also searches the two and three factor interactions that are linear/ nearly linear trend-free in the obtained linear trend free designs for main effects. This algorithm has been translated in Microsoft Visual C++ program. This program is general and using this, one can obtain the desired factorial experiment for any number of factors k (≥ 3). The catalogue of the obtained design for 2^k factorial experiment (for $k = 3, \ldots, 7$) that are linear trend-free for main effects along with two and three factor interactions that are linear trend-free are given.

In factorial experiment as the number of factors and/or levels increase, the number of treatment combinations becomes so large that it is not possible to accommodate them without losing homogeneity within block. To handle such situations the concept of confounding is introduced in factorial experiments. In a confounded factorial experiment, there is more than one block per replication and information on the factorial effects whose contrast becomes identical to the block contrast is lost. In confounded factorial experiments the experimental units within the blocks may exhibit trend effect over time or space. Thus algorithms to search/ generate confounded factorial experiments for which contrasts of main effects are free from linear trend and 2- and 3- factor that are linear trend-free/ nearly linear trend-free are also given in Chapter-II. These algorithms have been translated into Microsoft Visual C++ program. The algorithms along with the computer programs have been given in general for any number of factors each at two levels. The catalogues of confounded factorial experiments are prepared. These catalogues are restricted to factorial experiments for $k = 3$ to 7 factors each at 2 levels for confounded factorial experiments and has been presented in this chapter.

Chapter III is devoted to obtain trend-free fractional factorial plans. Algorithms have been developed to generate/ search fractional factorial designs that are linear trend-free for main effects and to identify two factor interactions that are linear trend free/ nearly linear trend-free using the criterion of complement foldover. For fractional factorial experiments, the algorithm along with the computer program in Microsoft Visual C++ has been given in general for any number of factors each at two levels. The catalogue has been prepared for $k = 5$ to 8 factors each at 2 levels.

Computer aided linear trend-free response surface designs are obtained in Chapter IV. In this chapter algorithm is developed to generate Box-Behnken response surface designs with the help of a given BIB design and makes a search for linear trend-free Box-Behnken from the obtained designs. The procedure is developed in general and the catalogue for linear trend-free Box-Behnken response surface designs is given for k taking values as 2, 3, 4 and 5.

The thesis is concluded with a list of references.

ABSTRACT

Designs for factorial experiments have been widely used in agricultural, biological and industrial experiments. The experimental units in a design of a factorial experiment may exhibit a trend over space or time. Such situations may occur in agricultural experiments when there is a slope in the field and there is sequential application of the treatments to the same experimental unit. This may also happen when the land is irrigated, the nutrients supplied by the fertilizers may not be equally distributed and trend in experimental units may be due to slope. When such land is irrigated, the nutrients supplied by the fertilizers may not be equally distributed and a slope may cause a trend in experimental units. In such type of experiments, the treatments are to be allocated to experimental units in some order to eliminate the effect of such trend. The resulting designs are called as *trend-free designs*.

In factorial experiments, the interest of the experimenter is to obtain the design in which contrasts of main effects and low order treatment combinations are estimated with free from linear trend when experimental material is influenced by trend effect. This research work is devoted to the development of computer algorithms for construction of trend-free designs for multifactor experimental settings. The algorithms developed are helpful in generating the complete and confounded factorial experiments that are linear trend-free for main effects. A search has been made to identify the two and three factor interactions that are estimable free or nearly free from linear trend effect. The algorithms are developed using the criterion of component-wise product. This algorithm has been translated in Microsoft Visual C++ program. From this program, one can obtain the desired factorial experiment for any number of factors k ($\geq$ 3) each at two levels. The catalogue of the obtained design for 2^k factorial experiment (for k = 3, …, 7) that are linear trend-free for main effects along with two and three factor interactions that are linear trend-free are given for without confounding and with confounding factorial experiments separately.

Algorithms have been developed to obtain fractional factorial designs that are linear trend-free for main effects and to identify two factor interactions that are linear trend

free/ nearly linear trend-free using the criterion of complement foldover. Algorithm is developed to obtain computer aided linear trend-free Box-Behnken response surface designs with the help of a given BIB design. The catalogue for fractional factorial plans for k = 5 to 8 factors each at 2 levels and for linear trend-free Box-Behnken response surface designs for k taking values as 2, 3, 4 and 5 have been prepared.

सार

कृशिविज्ञान, जीवविज्ञान एवं औद्योगिक परिक्षणों में फैक्टोरियल परीक्षण अभिकल्पनाओं का प्रायेाग बहुतायत में किया जाता है। फैक्टोरियल परिक्षणों के एकक भी समय व स्थान के अनुसार प्रवृत्ति प्रदर्शित कर सकते हैं। रेस्पांश वैरियेबल को ब्लॉक व ट्रीटमेण्ट के अलावा क्रमबद्ध प्रवृत्ति भी कुछ प्रायेागिक परिस्थितियो में प्रभावित कर सकती है। कृशि परीक्षणों में भी ऐसी परिस्थितियॉ आ सकती हैं। जैसे पहाड़ी क्षेत्रों में ढ़लान वाली उबाड़–खाबड़ जमीन। ऐसे क्षेत्रों की जब सिचाईं व उर्वरक दिया जाता है। तो पोशक तत्व पूरे क्षेत्र में समान रूप से वितरित नही होता है। तथा ढ़लान के कारण एक क्रमबद्ध प्रवृति देखी जा सकती है। ऐसी परिस्थितियों में ट्रीटमेण्ट का आकस्मिक प्रयोग न करके एक खास क्रम में देते हैं, ताकि क्रमबद्ध प्रवृत्ति का प्रभाव समाप्त किया जा सके। ऐसी अभिकल्पनाओं कों प्रवृत्तिमुक्त अभिकल्पना कहते हैं।

फैक्टोरियल परीक्षणों में प्रयोगकर्ता की रूचि ऐसी अभिकल्पनाओं में होती है जिसमे मुख्य कारक एवं लोअर ऑर्डर इन्टरैक्शन के मूल्यांकन रेखीय प्रवृत्तिमुक्त हो, जबकि परीक्षण एककों में रेखीय प्रवृत्ति हो। इस शोधकार्य में संगणक की सहायता से बहुकारक रेखीय प्रवृत्तिमुक्त अभिकल्पनाओं का अन्वेषण किया गया है। विकसित क्रमबद्ध निर्देशों की मदद से प्रवृति मुक्त कम्पलीट एवं फ्रैक्शनल फैक्टोरियल परिक्षणों को प्राप्त किया जा सकता हैं, जो मुख्य कारको के लिए प्रवृत्तिमुक्त होती है। रेखीय प्रवृत्तिमुक्त 2– एवं 3– फैक्टर इन्टरैक्शन का अन्वेषण किया गया हैं। फैक्टोरियल परिक्षणों के लिए ट्रीटमेण्ट कॉम्बीनेशन को प्रमाणिक क्रम में लिखते हैं। सभी मुख्य कारकों के सदिशों का कम्पोनेन्ट वाइज गुणा किया जाता है। प्राप्त सदिशों में यदि ″क″ सदीशों का कम्पोनेन्ट वाइज गुणांक है तो कारक ″क – 1″ क्रमांक के लिए प्रवृत्तिमुक्त हो जाता है। विकसित क्रमबद्ध निर्देशों की मदद से प्रवृत्तिमुक्त फैक्टोरियल परिक्षणों को प्राप्त किया जा सकता है। कन्फाउण्डिंग की धारणा का समावेश भी किया गया है। इससे एक से अधिक ब्लॉक में परिक्षण करने का विकल्प प्राप्त होता है। विकसित क्रमबद्ध निर्देशों की मदद से इसे प्राप्त कर सकते हैं। दी गई सूची में रेखीय प्रवृत्तिमुक्त फ्रैक्शनल फैक्टोरियल (क = 3, ..., 7) का समावेश है। फ्रैक्शनल फैक्टोरियल परिक्षण के लिए पहले डिफाइनिंग कॉट्रास्ट का निर्धारण कर लेते हैं। फिर विशम क्रमांक का डिफाइनिंग कॉट्रास्ट से छोटा ब्लॉक बनाकर, उसका फोल्डओवर कर प्रवृत्ति मुक्त अभिकल्पना प्राप्त की जा सकती है।

विकसित क्रमबद्व निर्देशों के अनुवाद माइक्रोसॉफ्ट विजुअल सी ++ में किया गया है, इसकी सहायता से इच्छित फ्रैक्शनल फैक्टोरियल (क $\geq \mathbf{3}$) प्राप्त किया जा सकता है जबकि प्रत्येक कारक के दो लेवेल हो। दी गई सूची में रेखीय प्रवृत्तिमुक्त फ्रैक्शनल फैक्टोरियल (क = 5, ..., 8) का समावेश है।

रिस्पॉन्स सरफेस अभिकल्पनाओ में बॉक्स–बेहेनकेन अभिकल्पनाओं के लिए रैखिय प्रवृत्तिमुक्त अभिकल्पनाओ का विकास किया गया तथा एक क्रमबद्व निदेशो तैयार किया गया है। जो प्रयोगकर्ताओं की मात्रात्मक कारकों के लिए प्रवृत्तिमुक्त रिस्पॉन्स अभिकल्पना उपलब्ध कराएगा। दी गई सूची में रेखीय प्रवृत्तिमुक्त बॉक्स बेहेनकेन अभिकल्पनाओं (क = 2, ..., 5) का समावेश है।

References

Bailey, R.A., Cheng, C.S. and Kipnis (1992). Construction of trend-resistant factorial designs. *Statist. Sinica*, **2**, 393-411.

Box, G.E.P. (1952). Multi-factor designs of first order. *Biometrika*, **39**, 49-57.

Box, G.E.P. and W.A. Hay (1953). A statistical design for the efficient removal of trends occurring in a comparative experiment with an application in biological assay. *Biometrika*, **9**, 304-319.

Box, G.E.P. and Behnken, D.W. (1960). Some new three level designs for the study of quantitative variables. *Technometrics*, **2**, 455-475.

Bradley, R.A. and Yeh, C.M. (1980). Trend-free block designs: Theory. *Ann. Statist.*, **8**, 883-893.

Bradley, R.A. and Odeh, R.E. (1988). A generating algorithm for linear trend free block designs. *Comm. Statist.-Simul. Comput.*, **17**, 1259-1280.

Chai, F.S. and Majumdar D. (1993). On the Yeh-Bradley conjecture on linear trend free block designs. *Ann. Statist.*, **21**, 2087-2097.

Chai, F.S. (1995). Construction and optimality of nearly trend-free designs. *J. Statist. Plan. Infer.*, **48**, 113-129.

Cheng, C.S. (1985). Run orders of factorial designs. *In Proceedings of the Berkeley Conference in Honor of Jerzy Neyman and Jack Kiefer* (L.M. Le Cam and R.A. Olshen, eds.), **2**, 619-633. Wadsworth, Monterey, California.

Cheng, C.S. and Jacroux, M. (1988). On the construction of trend-free run orders of two level factorial designs. *J. Amer. Statist. Assoc.*, **83**, 1152-1158.

Cheng, C.S. (1990). Construction of run orders of factorial designs. In *Statistical Design and Analysis of Experiments* (S. Ghosh, ed), 423-439. Dekker, New York.

Coster, D.C. and Cheng, C.S. (1988). Minimum cost trend-free run orders of fractional factorial designs. *Ann. Statist.*, **16**, 1188-1205.

Coster, D.C. (1993). Tables of minimum cost, linear trend-free run sequences for two and three-level fractional factorial design. *Compu. Statist. Data Analysis*, **16**, 325-336.

Coster, D.C. (1993). Trend-free run orders of mixed-level fractional factorial designs. *Ann. Statist.*, **21**, 2072-2086.

Cox, D.R. (1951). Some systematic experimental designs. *Biometrika*, **38**, 312-323.

Cox, D.R. (1958). *Planning of experiments.* John Wiley & Sons, New York.

Daniel, C. and Wilcoxon, F. (1966). Factorial 2^{n-p} plans robust against linear and quadratic trends. *Technometrics*, **8**, 259-278.

Dhall, S.P. (1986). Some studies on robustness of designs, *Unpublished Ph.D. Thesis*, IARI, New Delhi.

Dickinson, A. W. (1974). Some run orders requiring a minimum number of factor level Changes for the 2^4 and 2^5 main effects plans. *Technometrics*, **16**, 31-37.

Draper, N.R. and Stoneman, D.M. (1968). Factor changes and linear trends in eight-run two-level factorial designs. *Technometrics*, **10**, 301-311.

Dwivedi, S.K. (1997). Computer aided search for optimal designs. *Unpublished Ph.D. Thesis*, IARI, New Delhi.

Federer, W.T. and Schlottfeldt, C.S.(1954). The use of covariance to control gradients in experiments. *Biometrics*, **10**, 282-290.

Hill, H.M. (1960). Experimental designs to adjust for time trend. *Technometrics*, **2**, 67-82.

Hinkelmann, K. and Kempthorne, O. (1994). *Design and Analysis of Experiments. Vol. I: Introduction to Experimental Design*. New York, NY: John Wiley and Sons, Inc.

Hinkelmann, K. and Jo, Jinnam (1998). Linear trend-free Box-Behnken Designs. *J. Statist. Plan. Infer.*, **72**, 374-354.

Jacroux, M. and SahaRay, R. (1990). On the construction of trend-free row-column 2-level factorial experiments. *Metrika*, **37**, 163-180.

Jacroux, M. Majumdar, D. and Shah, K.R. (1995). Efficient block designs in the presence of trends. *Statistica Sinica*, **5**, 605-615.

Jacroux, M., Majumdar, D. and Shah, K.R. (1997). On the determination and construction of optimal block designs in the presence of linear trends. *J. Amer. Statist. Assoc.*, **92**, 375-382.

Jan, H.W. and Wang, P.C. (1995). Designing two-level factorial experiments using orthogonal arrays when the run order is important. *The Statistician*, **44**, 379-388.

Jo, Jinnam and Hinkelmann, K. (1993). Some properties of Box-Behenken Designs. *J. Combin. Inform. System Sci.*, **18**, 273-287.

John, P.W.M. (1990). Time trend and factorial experiments. *Technometrics*, **32**, 275-282.

Lal, K. Parsad, R. and Gupta, V.K. (2005). A study on trend-free designs. Project Report, IASRI, New Delhi

Kohli, P. (2006). A study on supersaturated designs. *Unpublished M. Sc. Thesis*, IARI, New Delhi.

Majumdar, D. and Martin, R. J. (2002). Finding optimal designs in the presence of trends. *J. Statist. Plan. Infer.*, **106**, 177-190.

Nguyen, N.K. (1983). Computer aided construction of optimal designs. *Unpublished Ph.D. Thesis*, IARI, New Delhi.

Philips, J.P.N. (1964). The use of magic squares for balancing and assessing order effects in some analysis of designs. *Appl. Statist.*, **13**, 1367-73.

Philips, J.P.N. (1968a). A simple method of constructing certain magic rectangles of even order. *Math. Gazette*., **52**, 9-12.

Philips, J.P.N. (1968b). Methods of constructing one-way factorial designs balanced for trend. *Appl. Statist*., **17**, 3857-3863.

Rathore, A. (2004). Development of algorithms for computer aided search for optimal/nearly optimal designs. *Unpublished Ph.D. Thesis*, IARI, New Delhi.

Rathore, A. Parsad, R. and Gupta, V.K. (2004). Computer aided construction and analysis of augmented designs. *J. Indian Soc. Agricultural Statist.* **57**, Special Volume, 320-344.

Sachdev, A.K., Ahuja, S.D. and Ram Gopal (1989). Feed consumption, egg production and egg quality traits as influenced by cage-tier locations of Japanese quail. *Ind. J. Anim. Sci.* **59 (7)**, 860-865.

Satpati, S.K, Parsad, R., Gupta, V.K. and Nigam, A. K. (2006). Computer-aided search of efficient nested incomplete block designs for correlated observations. *J. Comb. Inf. Syst. Sci.* **31**, no. 1-4, 163-186.

Satpati, S.K. (2006). Computer aided search of efficient designs for dependent observations. *Unpublished Ph.D. Thesis*, IARI, New Delhi.

Yeh, C.M. and Bradley, R.A. (1983). Trend free block designs: existence and construction results. *Comm. Statist.-Theory Methods*, **12**, 1-21.

Yeh, C.M., Bradley, R.A. and Notz, W.T. (1985). Nearly trend free block designs. *J. Amer. Statist. Assoc.*, **80**, 985-992.