2011

Simulation Factor Screen in Binary Response Models

Minqi Li
West Virginia University

Simulation Factor Screen in Binary Response Models

Minqi Li

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Industrial Engineering

Feng Yang, Ph.D., Chair
Hong Wan, Ph.D.
Majid Jaraiedi, Ph.D.

Department of Industrial and Management Systems Engineering

Morgantown, West Virginia
2011

Keywords: Simulation, Factor Screen, Binary response model, Sequential test, Sequential bifurcation, Modified CSB, SFD-MT

ABSTRACT

Simulation Factor Screen in Binary Response Models

Minqi Li

To eliminate unimportant factors so that the remaining important factors can be further studied in later experimentation, screening experiments (which may be physical or simulation based) are typically performed. This thesis proposes a hybrid statistical procedure for efficient factor screening via simulation experiments. The hybrid procedure is particularly developed for cases where the system response is binary, as opposed to continuous; such a factor-screening procedure does not exist yet in the literature.

The proposed hybrid procedure integrates two screening methods: the sequential factorial design with multivariate sequential test (SFD-MT), which is newly developed in this work, and the modified controlled sequential bifurcation (CSB), which is adapted from the existing CSB method. At the beginning of the procedure, a pre-screening process is conducted to obtain the preliminary estimates of factor effects, and to determine whether SFD-MT or modified CSB will be used for factor screening. Then the selected screening method (either SFD-MT or modified CSB) are performed to identify the important factors based on simulation experiments. In both SFD-MT and CSB, the type I and type II errors are approximately controlled through appropriate hypothesis tests. The efficiency of the hybrid procedure over the CSB in the literature is demonstrated via empirical experiments.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Feng Yang, and Dr. Hong Wan in Purdue University, for their great guidance, for giving me the opportunity to work on this project, which could not have been written without their encouragement and support. I am also thankful to Dr. Majid Jaraiedi's serving on my committee, and for his insightful ideas and assistance in preparing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Factor screening is usually the first phase of experiment to investigate a system [1]. As the complexity of system increases, there are usually a large number of factors included while only a few of them are really important [2][3][4]. For instance, in the nuclear waste disposal problems, the simulation model has up to 3000 factors while less than 1% of the total are important [5][6][1]. Thus, factor screening is implemented to identify the important factors that have a significant impact on the system performance at the initial stage, then further experiments can be conducted on these important factors which may have interactions or higher order effects [1].

Trocine [1] has stated that there are mainly three criteria for a good factor screening procedure: effectiveness, efficiency and robustness. Effectiveness means that the screening procedure is able to find the true important factors. However, since the system being investigated are unknown to us, the effectiveness is difficult to measure directly. The second criteria is efficiency which is judged by the number of experiment runs needed for the procedure. The efficiency of a screening method depends on the size of the problem and also the underlying data structure from the system response. The third criteria is the robustness. A robust screening method means that it requires little prior information of the system and can be used in a wide area.

To meet the three criteria, good experiment design is needed in a factor screening procedure. The most commonly used experimental design is the fractional factorial design [7].

However, the number of experiment runs needed is huge when it is used in large size problem. To conduct factor screening on large-scale cases, a number of screening methods have been developed in the past thirty years. Trocine [1] made a complete overview of screening methods available for more than 20 factors such as the two-stage group screening method [8] [9][10], sequential bifurcation [2], super saturated design [11] and Trocine screening method [1].

In this thesis work, simulation experiments will be used to generate data for the system of interest. As indicated by Wan [3], simulation experiments have many advantages over physical experiments. Physical experiments usually deal with less than 20 factors and each experiment run costs considerable time and money. While simulation is usually cheaper and faster and can deal with a large number of factors that are impossible for physical experiments. Also, it is more convenient to adjust factor settings in the simulation experiments. In recent years, simulation has been widely used in operational research area. For example, in semiconductor industry, since the manufacturing process is very complicated: mixed types of products exist and each type of product may share common resources, simulation becomes a powerful tool to locate the bottleneck and improve the performance of the system.

1.2 Problem Statement and Objective

All the simulation screening methods proposed in the existing works only handle linear models with normally distributed error term, while in the real word categorical responses are commonly encountered as well. The binary response is the most common case of the categorical models and logistic regression is usually used to analyze the binary experiment results. Since the assumption of normal response is not meet in our case, many of the proposed experimental designs can not be used. In this thesis work, a new screening method is proposed which takes a hybrid framework [12] and combines two simulation screening procedures: Sequential Factorial Design with Multivariate Sequential Test (SFD-MT) and modified CSB [3]. At the pre-screen stage, the new method uses fractional factorial design

and logistic regression to get a rough estimation of the factor effects. Then factors are divided into potential important group and potential unimportant groups based on which the appropriate screening method is selected.

The objective of this work is to develop an efficient screening procedure by combining an appropriate sequential test and experimental design techniques to meet the error control requirement. There are two major challenges involved in this work. First, the proper screening framework and the multivariate sequential test method used in the SFD-MT need to be determined. Secondly, CSB needs to be modified such that it can be used into the binary response model. Since CSB was originally designed for linear model with normal errors, the binary response for experiments must be transformed properly to be asymptotically normally distributed.

This thesis work is organized as follows: chapter 2 reviews the existing work and introduces the screening method that will be used here. Chapter 3 describes the model we used and some notations in this work. Chapter 4 gives the details of SFD-MT and modified CSB as well as the performance of the hybrid method. Chapter 5 presents the empirical results and compares the hybrid method with SFD-MT and CSB. Chapter 6 provides a conclusion of this thesis work.

Chapter 2

Literature Review

In this chapter, screening methods proposed in the existing works are presented. Some of them focus on the efficiency and require many prior information of the systems, while others focus on the effectiveness and are required to control the error at user-specified level.

Watson [8] proposed the method of two-stage group screening which partitioned factors into groups by prior knowledge. In the first stage, all factors in the same groups are set to the same level such that each group can be dealt with like single factor. Then fractional factorial design is conducted on groups to test the group importance. In the second stage, factors in the important groups are partitioned into individual ones and fractional factorial design is run again on these factors to identify the important factors. The method is highly efficient since the number of the simulation runs required is usually only slightly higher than the number of factors. However, many assumptions are needed before experiment can be conducted. Mauro [9][10] demonstrated performance of the two-stage group screening method with empirical study. His works relaxed the assumptions and allowed unknown directions of effects and existence of interaction effects. His work has also shown the effect of design parameters on the performance of the two-stage group screening method.

Bettonvil and Kleijnen [2] developed sequential bifurcation (SB) which deals with the deterministic model. Cheng [13] expanded the SB into stochastic case where normal error with constant variance exists. SB requires known direction of factor effects and the same sign of main and interactive effects. The screening process is actually testing the effect of

factor groups, which is the sum of the effect coefficients. The idea is quite straight forward: if a group is unimportant, then there are no important factors included so all factors in that group should be eliminated; if a group is important, then there may be some important factors contained in the group and the group should be divided into two subgroups and further experiment run is conducted. This bifurcation terminates when there is no important groups can be separated. SB is highly efficient if important factors are clustered in the factor list. Thus, if there is some prior information, factors can be assigned positions appropriately before the screening procedure being conducted [13].

Trocine [1] introduced a new screening method called Trocine Screening Procedure which uses a three-stage experimental design. In the first stage, 3 replicates are run by setting all factors to higher level to estimate the range of the experiment region; in the second stage, several numbers of simulation runs determined by the number of factors are run. These runs are designed such that no factors are positively aliased with each other. In the third stage, new design points are generated based on a genetic algorithm. Finally, factors are ranked by scores and the top 25% of factors are selected as important ones.

Holcomb [11] investigated the contrast distribution in the super saturated design (SSD). The design matrix was partitioned for important factors and unimportant factors randomly since no prior knowledge of the system existed. It stated that the distribution of the contrast could be approximated by normal distribution. In the simulation study, several previous proposed SSD method based on contrast were compared in the linear model with constant coefficient and varying coefficient. Holcomb mentioned that the SSD is just a pre-screen method that eliminated a large portion of unimportant factors and the efficiency of SSD depends on the number of unimportant factors in the group and the prior information available for the system.

All of these methods assume linear model with equal variance or even zero variance and the efficiency of them are extremely high in certain cases. However, the disadvantage is that many of the assumptions used can not be meet in the real word such that the

performance of these methods can not be guaranteed. The Controlled Sequential Bifurcation (CSB) by Wan et al. [3] and Controlled Sequential Factorial Design (CSFD) by Shen and Wan [14] are designed to control both the type I and type II error and require less prior information of the system. These two methods can deal with linear models with heterogenous variance and provide desired error control. CSB has combined the hypothesis test developed by Kim [15] with sequential bifurcation procedure proposed by Bettonvil and Kleijnen [2]. It requires known direction of factor effects and assumes that there are only main effects in the model. The framework of CSB is the same as the sequential bifurcation but the effect of groups are tested by sequential test to control the error. CSB-X by Wan et al. [16] improves the efficiency and efficacy of CSB by incorporating a fold-over design to deal with the interaction effects that exists in the model.

CSFD generates random observations in batches by fractional factorial design and tests factor effects one by one. The procedure first generates a number of observations to get an initial estimation of factor effects, then more observations will be generated during the procedure if no conclusion has been reached out. All random observations can be used during screening procedure and the importance of interaction effects can also be tested if fractional factorial designs with resolution IV or V are used.

2.1 SFD-MT and Modified CSB

All the screening methods introduced above assume that the response of the model is normally distributed. To deal with the binary response case, we introduce two new methods: SFD-MT and Modified CSB.

SFD-MT combines experimental design with a sequential hypothesis test method with multiple endpoints. The way SFD-MT deals with observations is similar to that in CSFD: observations are generated in batches and we call each batch as one replication [14]. Within each replication, observations provide one estimate for effect coefficients by using Maximum likelihood Estimation (MLE) [17]. The importance of factor effects are determined based on

these replications. Since the estimates of effect coefficients by MLE usually have correlations with each other, it is not appropriate to test the factor effects one by one as in CSFD. Thus the SFD-MT tests the importance of group of factors based on the estimates of coefficients and the correlations of the estimates. Besides, a sequential bifurcation frame is used in the procedure. Since some estimates of factor effect coefficients have high correlation with each other, we are likely to put those factors into the same subgroup when splitting. Replications generated in different splitting stages can be used during the entire screening procedure. SFD-MT can work with binary response model or linear model with highly heterogenous variance.

Since estimates of effect coefficients have correlation with each other, we would like to test the importance of factor effects in group. Sequential hypothesis test with multiple endpoints is needed in our screening procedure. Jennison and Turnbull [18] proposed multiple sequential test based on the exact distribution of test statistics. The test requires that the covariance matrix can be written in the form of a known matrix and a scalar. However, in a binary response model, the covariance matrix of estimates of effect coefficients is usually completely unknown. Tang et al. [19] developed a sequential test based on the test proposed by O'Brien [20]. This sequential test can be applied to the situation when covariance matrix is completely unknown. However, when the number of factors is large, the convergency of the asymptotically normal distribution will be slow and affects the performance of the test. The sequential test used in SFD-MT was proposed by Jackson and Bradley [21]. There is no requirement on the covariance matrix and the direction of the factor effects. A heuristic method was prompted to modify the test such that it can be used here. It can be shown that that type I error can be controlled during the entire screening procedure and the performance of power control will be evaluate in the empirical study.

Modified CSB has the same splitting framework of the CSB but it deals with different responses within each splitting stage. Original CSB is used in linear model with normal error terms, so the test statistics used in every splitting stage is strictly normally distributed. On

the other hand, modified CSB is dealing with binary response model and the log odds ratio is used to test the group effects. By large sample theory, log odds ratio is asymptotically normally distributed under certain constraints. So if we design the experiment appropriately, it can be treated as a normally distributed random variable and many sequential tests can be used to control the type I error and the power in our case.

Chapter 3

Model Description

Suppose there are p factors to be tested in the group. Let $\boldsymbol{\beta} = (\beta_1, ..., \beta_{p-1}, \beta_p)^T$ be effect coefficients and $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_1, ..., \widehat{\beta}_{p-1}, \widehat{\beta}_p)^T$ be the estimator of $\boldsymbol{\beta}$. $\widehat{\boldsymbol{\beta}}$ can be calculated from a logistic regression process and is asymptotically normally distributed as $\widehat{\boldsymbol{\beta}} \sim \mathbf{N}(\boldsymbol{\beta}, \boldsymbol{\Sigma})$ under certain condition. The covariance matrix $\boldsymbol{\Sigma}$ is usually unknown but can be estimated by sample covariance matrix $\widehat{\boldsymbol{\Sigma}}$. Binary response model is considered here and we assume that there is no interaction effects between two factors. Suppose that the response variable Y satisfies binomial distribution with $Pr(Y = 1) = \pi(\boldsymbol{x})$ and $Pr(Y = 0) = 1 - \pi(\boldsymbol{x})$, then we can investigate it by logistic regression. Let the Logit link function be $\eta = logit(\pi(\boldsymbol{x}))$, then η has the following linear form:

$$log(\frac{\pi(\boldsymbol{x})}{1 - \pi(\boldsymbol{x})}) = \eta = \beta_0 + \sum_{i=1}^{p-1} \beta_i x_i$$

Here $\boldsymbol{x} = (x_1, x_2, ..., x_p)$ are level settings and $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)^T$ are effect coefficients. The β_0 is the intercept terms which is not of our interest.

The objective of screening procedure is to classify factors into important ones and unimportant ones with controlled type I error and type II error. Typically in factor screen, type I error means that an unimportant factor is classified as important; type II error means that an important factor is classified as unimportant. Suppose that we have two thresholds, Δ_0 and Δ_1 which satisfy $\Delta_0 < \Delta_1$. Each factor effect is compared with the two thresholds. For $i = 0, 1, 2, ..., p - 1$, when $|\beta_i| < \Delta_0$, we should have probability $\leq \alpha$ to classify factor i as important; when $|\beta_i| > \Delta_1$, we should have probability $\leq \gamma$ to classify factor i as

unimportant, when $\Delta_1 \geq |\beta_i| \geq \Delta_0$, we expect the screening procedure to classify them as important but not guarantee to do that [14]. Wan et al. [3] proposed a cost model to determine the thresholds and factor effects such that all the factor effects can be compared in the same standard. After some minor change, the cost model is good to be implemented into this screening procedure. Here are notations that will be used in this paper:

$\circ$ p: Total number of factors in the group.

$\circ$ α: The probability to make type I error.

$\circ$ γ: The probability to make type II error.

$\circ$ $\triangle_0$: Lower Threshold.

$\circ$ $\triangle_1$: Higher Threshold.

$\circ$ N_0: Initial number of replications generated in SFD-MT or modified CSB.

Notations used in SFD-MT:

$\circ$ l: Number of observations at each design point for logistic regression.

$\circ$ $Y(i)$: The random observation in the ith replication.

$\circ$ $\widehat{\boldsymbol{\beta}}(i)$: estimate of $\boldsymbol{\beta}$ from the ith replication.

$\circ$ $\widehat{\beta}_j(i)$: estimate of β_j from the ith replication.

$\circ$ $B(n) = \frac{1}{n}\sum_{i=1}^{n}\widehat{\boldsymbol{\beta}}(i)$ is the average of estimated effect coefficients from n replications.

$\circ$ $\widehat{\boldsymbol{\Sigma}}(n) = \frac{1}{n}\sum_{i=1}^{n}\frac{(\widehat{\boldsymbol{\beta}}(i)-B(n))(\widehat{\boldsymbol{\beta}}(i)-B(n))^T}{n-1}$ is the sample covariance matrix estimated from n replications.

Notations used in modified CSB:

$\circ$ l_c: number of observations within each replication.

$\circ$ $m_k(i)$: The number of successes in the ith replication at level k.

$\circ$ $LO_k(i)$: The log odds ratio calculated from the ith replication at level k.

$\circ$ $D(k_2,k_1)(j)$: $D(k_2,k_1)(j) = \frac{1}{j}\sum_{i=1}^{j}(LO_{k_2}(i) - LO_{k_1}(i)), j = min(n_{k_1}, n_{k_2})$ is the average of the difference of log odds ratio of level k_1 and level k_2 from the first j replications.

Chapter 4

Methodology

The hybrid methodology proposed by Shen et al. [12] is used here to conduct the simulation-based factor screening. The idea is straight forward since both modified CSB and SFD-MT have drawbacks when dealing with certain factor groups. Modified CSB is highly efficient only when the number of potential important factors (factor effect coefficient greater than Δ_0) is small and these factors are clustered. SFD-MT is inefficient when deals with a large number of factors.

The hybrid method contains two phases. In phase I, fractional factorial design considering all factors is conducted to get a rough estimate of all factor effect coefficients. Then based on the prescreening results, factors are divided into three groups based on a user-specified threshold $\triangle_t > 0$, called splitting threshold. Assume that the estimate of coefficient β_i in the prescreeng is $\widehat{\beta}_i$. Then for the ith factor, if $|\widehat{\beta}_i| \geq \triangle_t$, then it will be put into the first group, called IMP group; if $0 \leq \widehat{\beta}_i < \triangle_t$, it will be put into the second group, called P-UNIMP group; if $0 < \widehat{\beta}_i < \triangle_t$, it will be put into the third group, called N-UNIMP group [12]. The factors in the P-UNIMP and N-UNIMP are ordered based on the absolute values of their effect coefficients which are estimated in the prescreen. Since the number of important factors are assumed to be small, the size of the IMP group that contains potentially important factors is expected to be relatively small. On the other hand, the P-UNIMP and N-UNIMP groups that contain potentially unimportant factors should have a larger size than the IMP group and the potentially important factors, if there is any, should be clustered after the ordering. In phase II, the three groups are screened by

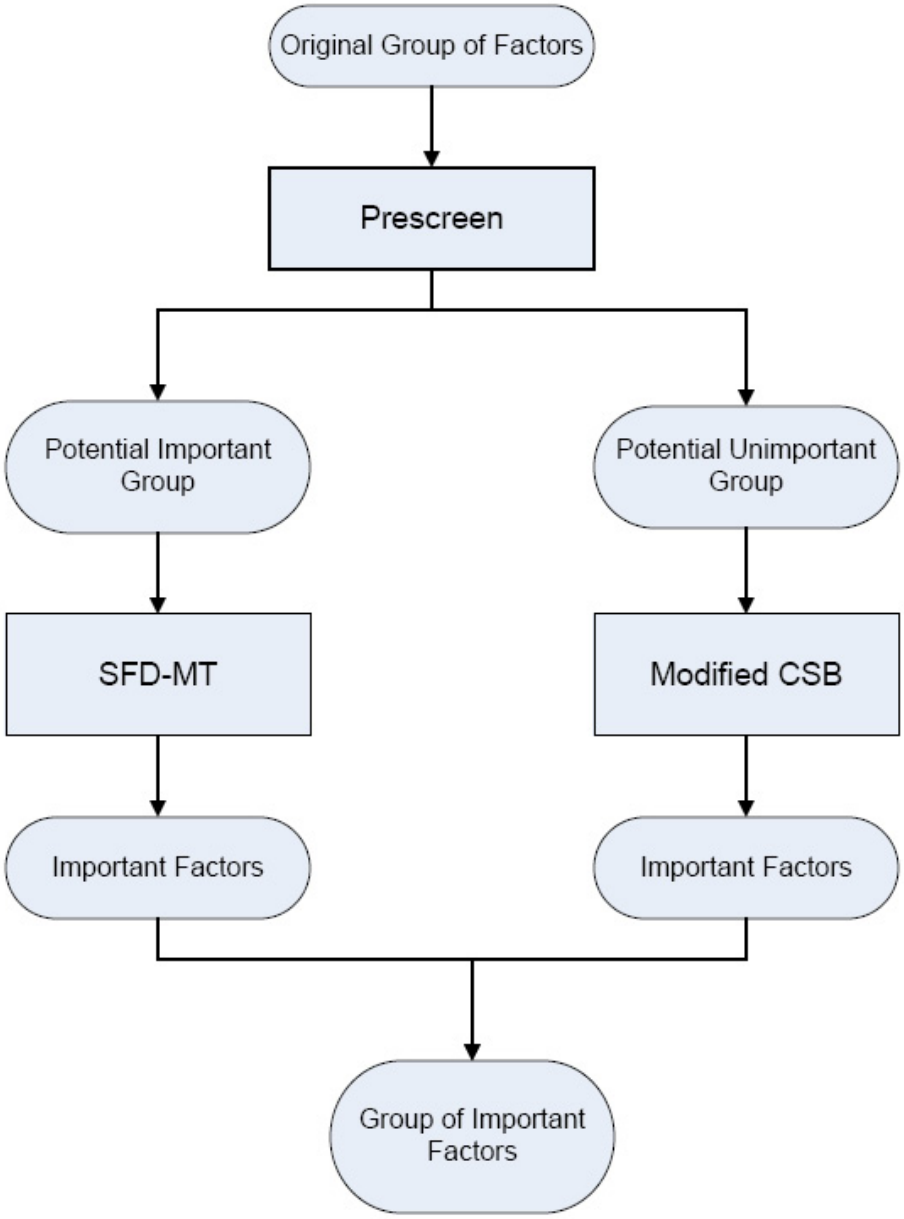

Figure 4.1: The flowchart of the screening procedure

appropriate procedure. SFD-MT is implemented to screen the IMP group and the modified CSB, which will be introduced in the later section, is implemented to screen the P-UNIMP and N-UNIMP group. The flowchart 4.1 shows the framework of the hybrid method. We organize this chapter as follows: section 4.1 will introduce the SFD-MT procedure; section 4.2 tells the detail of modified CSB; section 4.3 talks about the error control of this hybrid

method. For all the experimental design used in our work, we assume the factor have two levels: 1, -1, which is the higher level and lower level respectively.

4.1 SFD-MT

In SFD-MT, we generate random observations in batches to estimate effect coefficients and each batch is called a replication. Since only main effects exist, resolution III fractional factorial design is used within each replication. As indicated in Montgomery [7], if there are p main effects, we can find an m satisfying $2^{m-1} \leq p < 2^m$ to construct resolution III fractional factorial design. Since observations are independent with each other in different replications, the estimate from each replication is also independent with each other. In this section, we assume that the group being screened by SFD-MT contains p factors $\beta_1, \beta_2, ..., \beta_p$. The general framework of SFD-MT is similar to CSB [3], which is shown bellow:

Initialization: Generate N_0 replications, calculate $\widehat{\boldsymbol{\beta}}(N_0)$, $\widehat{\boldsymbol{\Sigma}}(N_0)$ and set sample size n equal to N_0. Assign index i to β_i for $i = 1, 2, ...p-1, p$. Create an empty FIFO queue. Construct a group that contains all the factors and put this group into the FIFO queue. All groups are tagged by their orders of entering the FIFO queue. For instance, the group that contains all factors is tagged by index 1 since it is the first one which enters the FIFO queue.

While FIFO queue is not empty, do

Remove a group from the queue and test the importance of the group effect. Generate more replications if needed and update sample size n. If the test result is:

Important and group size$= 1$: The only factor in the group is classified as important.

Important and group size> 1: Split it into two subgroups based on sample covariance matrix $\widehat{\boldsymbol{\Sigma}}(n)$ and then add each subgroup into the FIFO queue.

Unimportant: Classify all the factor in the group as unimportant.

End

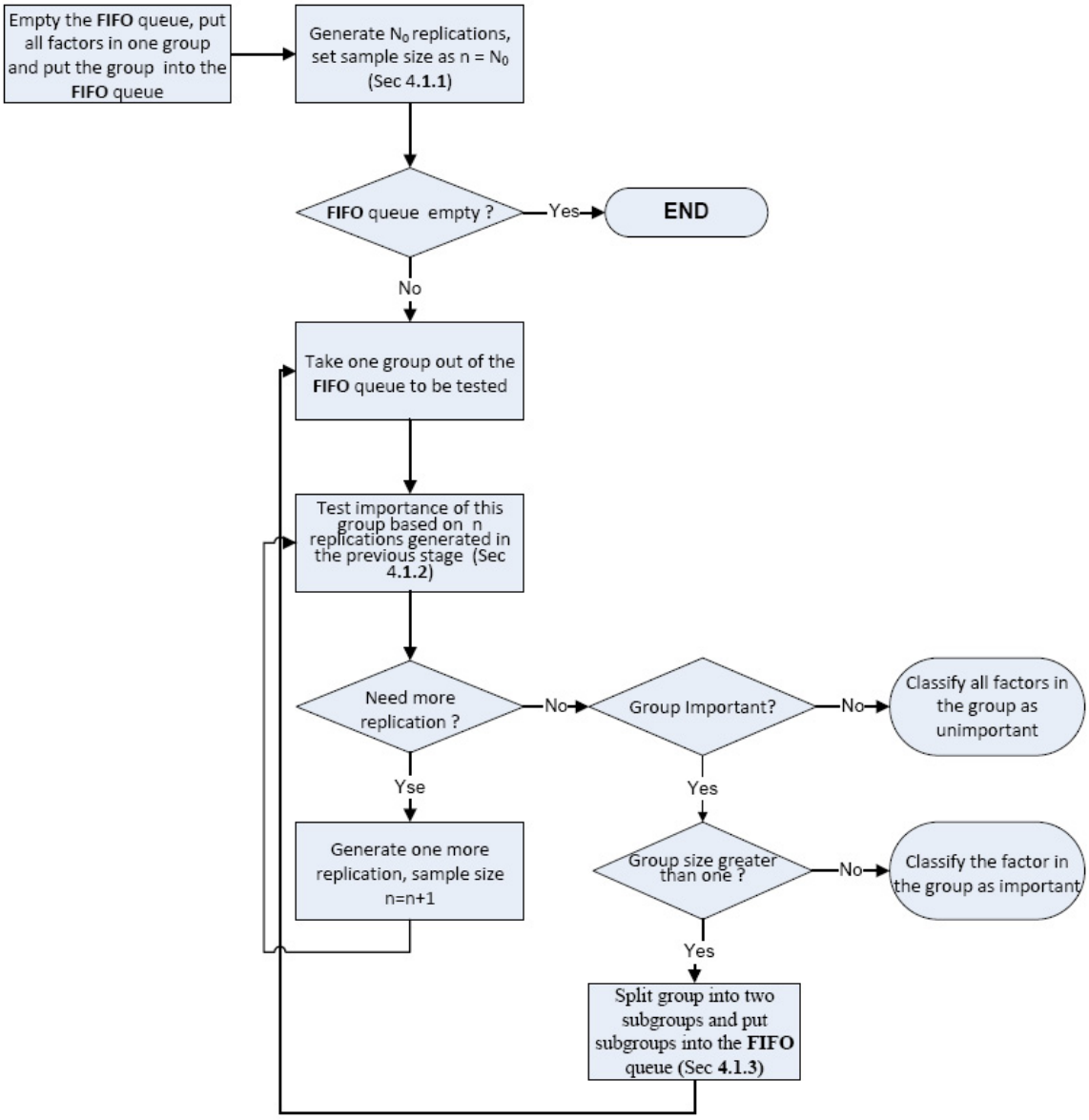

Figure 4.2: The flowchart of the SFD-MT

Flowchart in figure 4.2 gives an overview of how the SFD-MT works. In section 4.1.1, we will talk about the experimental design used and how we calculate the estimate of effect coefficients from observations. The details of implementing the sequential test into screening procedure is introduced in section 4.1.2. The group splitting process is introduced in section 4.1.3. The performance of the SFD-MT is discussed in section 4.1.4.

4.1.1 Replications Generation

Resolution III fractional factorial design is used to generate replications. If we have p factors totally, then there exists an m less than p such that $2^{m-1} \leq p < 2^m$ and corresponding fractional factorial design can be conducted [7]. In logistic regression, effect coefficients are estimated by maximum likelihood estimation. Let M be the total number of design points and l_i be the number of observations at each design point, $i = 1, 2, ..., M$, so within each replication, there are totally $\Sigma_{i=1}^{M} l_i$ observations. The number of observations at each design point need to be set large enough to guarantee the convergency of the iterative weighted least square algorithm when fitting the model. The details of MLE for logistic regression can be found in McCullagh [17] and have been put it in Appendix.

The sequential hypothesis test used here requires normal distribution of $\widehat{\boldsymbol{\beta}}(i), i = 1, 2, ..., n$. Beer [22] has proved that as the number of design points, M, grows large, the MLE will be asymptotically normally distributed. Usually when M is large, sparsity always exists, which means there exists some l_i that equals to 1. But we do not need to worry about this problem in simulation experiments. If the number of factors is large, we usually have a M large enough to guarantee the asymptotically normality of maximum likelihood estimator.

4.1.2 Test Procedures

Assume that at certain splitting stage of SFD-MT, we are testing the group k which is the kth group entering the FIFO queue. Group k contains m factors and in the previous stage, n replications have already been generated ($n \geq N_0$). The index of the m factors in group k is $(i_1, i_2, ...i_m)$ which satisfies $0 \leq i_j \leq p, \mathrm{j} = 1, 2, ..., m$ and $i_1 < i_2 < ... < i_m$. If we write $B(n) = (b_1(n), b_2(n), ...b_p(n))$, $\boldsymbol{\Sigma} = (a_{ij})_{p\times p}$ and $\widehat{\boldsymbol{\Sigma}}(n) = (\widehat{a}_{ij}(n))_{p\times p},\ i, j = 1, 2..., p$, then the estimate of effect coefficients and sample covariance matrix used in current stage for group k is $B_k(n) = (b_{i_1}(n), b_{i_2}(n), ..., b_{i_m}(n))$ and $\widehat{\boldsymbol{\Sigma}}_k(n) = (\widehat{a}_{i_l i_h}(n))_{m\times m},\ l, h = 1, 2, ..., m$. If we write $\boldsymbol{\beta}^k = (\beta_{i_1}, \beta_{i_2}, ..., \beta_{i_m})^T$ and $\boldsymbol{\Sigma}_k = (a_{i_l i_h})_{m\times m},\ l, h = 1, 2, ..., m$, we do the hypothesis

test

$$H_0 : (\boldsymbol{\beta}^k)^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\beta}^k = \lambda_0^k(n)^2$$

against the alternative

$$H_a : (\boldsymbol{\beta}^k)^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\beta}^k = \lambda_1^k(n)^2$$

The rule of choosing $\lambda_0^k(n)^2$ and $\lambda_1^k(n)^2$ will be talked in section 4.14. The ratio test procedure developed by Jackson and Bradley [21] is shown bellow :

Initialization: n replications have been generated in previous stages ($n \geq N_0$). We calculate

$$T_k^2(n) = n(B_k(n))^T \widehat{\boldsymbol{\Sigma}}_k(n)^{-1} B_k(n)$$

$$Ratio_k(n) = \frac{p_{1k}(n)}{p_{2k}(n)} = exp(-n(\lambda_1^k(n)^2 - \lambda_0^k(n)^2)/2) \frac{{}_1F_1(n/2, k/2; n\lambda_1^k(n)^2 T_k^2(n)/2(n-1+T_k^2(n)))}{{}_1F_1(n/2, k/2; n\lambda_0^k(n)^2 T_k^2(n)/2(n-1+T_k^2(n)))}$$

While $\gamma/(1-\alpha) \leq Ratio_k(n) \leq (1-\gamma)/\alpha$ **do**

Generate the $n+1th$ replication and replace n by new value $n+1$. Then update the value of $B(n), B_k(n), \widehat{\boldsymbol{\Sigma}}(n), \widehat{\boldsymbol{\Sigma}}_k(n), \lambda_0^k(n)^2, \lambda_1^k(n)^2, T_k^2(n)$ and $Ratio_k(n)$.

End

If $Ratio_k(n) > (1-\gamma)/\alpha$ **Then** classify group k as important.

If $Ratio_k(n) < \gamma/(1-\alpha)$ **Then** classify group k as unimportant.

4.1.3 Splitting Process

The splitting process is implemented based on the sample covariance matrix. Assume that group k containing m factors has been classified as important. All the notations used are the same as in section **4.1.2**. Also assume that n replications have been generated and $\widehat{\boldsymbol{\Sigma}}_k(n) = (\widehat{a}_{i_l i_h}(n))_{m \times m}, \mathrm{l}, h = 1, 2, ..., m$. We want to split the group into two subgroups. If m is even, each subgroup has $m/2$ factors; if m is odd, one subgroup has $[m/2]+1$ factors and the other has $[m/2]$ factors. Since the higher the correlation between two estimates of effect coefficients is, the more likely the two estimates would "affect" each other. So our

general idea is to put factors into the same group if estimates of their effect coefficients have high correlations. The detailed splitting process is described as follows:

Step 0. Creat an empty remaining list.

Step 1. Rank the absolute value of $\widehat{a}_{i_l i_h}(n),\ l > h,\ l, h = 1, 2, ..., m$ and put them into a remaining list.

Step 2. If $\widehat{a}_{i_{l_0} i_{h_0}}(n)$ has the largest absolute value, put factor i_{l_0} and i_{h_0} into subgroup 1 and remove $\widehat{a}_{i_{l_0} i_{h_0}}(n)$ from the remaining list.

Step 3. Find the one in the remaining list that has the largest absolute value, if one of its two index is already in subgroup 1, we put the factor with the other index into subgroup 1; otherwise we put both of them into subgroup 1.

Step 4. Check the number of factors in subgroup 1. When m is odd, if the number of factors in subgroup 1 is less than $[m/2]$, return to step 3; otherwise put all other factors that are not in subgroup 1 into subgroup 2 and the splitting process is completed. When m is even, if the number of factors in subgroup 1 is less than $m/2$, return to step 3, else if the number of factors in group 1 is greater than or equals to $m/2$, put all other factors not in subgroup 1 into subgroup 2 and the splitting process is completed.

4.1.4 Performance of SFD-MT

The SFD-MT is supposed to control type I error and type II error when we do the hypothesis

$$|\beta_i| < \Delta_0 \text{ against } |\beta_i| \geq \Delta_0$$

Given a $p-$dimension multivariate normal population $\boldsymbol{y}$ with covariance matrix $\boldsymbol{\Sigma}$, the ratio test can handle the following composite hypothesis and control type I error α and type II error γ:

$$H_0 : \boldsymbol{\beta}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta} = \lambda_0^2 \text{ against } H_1 : \boldsymbol{\beta}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\beta} = \lambda_1^2$$

The form of this composite hypothesis is different from the previous one. The test satisfies that if $\boldsymbol{\beta}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\beta}$ is less than or equal to λ_0^2, it has the probability $1-\alpha$ to accept the null hypothesis; if $\boldsymbol{\beta}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\beta}$ is greater than or equal to λ_1^2, it has the probability $1-\gamma$ to reject the null hypothesis; if $\boldsymbol{\beta}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\beta}$ is greater than λ_0^2 but less than λ_1^2, the test should reject the null hypothesis but not guarantee to do that. As indicated by Jackson and Bradley [21], there exists many ways to specify λ_0^2 and λ_1^2 when applying this test to real cases while each case needs to be handled individually. In our case, we will select the two thresholds in a heuristic way.

Suppose $\boldsymbol{\Sigma^{-1}} = (\sigma_{ij}^{-1})_{p\times p}$ is the inverse of the covariance matrix. Then we have $\boldsymbol{\beta^T\Sigma^{-1}\beta} = \sum_{i=1}^{p}\sigma_{ii}^{-1}\beta_i^2 + \sum_{i=1}^{p}\sum_{j<i}2\sigma_{ij}^{-1}\beta_i\beta_j$. Our goal is to compare the group effect, which is the sum of effect coefficients, with the two threshold, but $\boldsymbol{\beta^T\Sigma^{-1}\beta}$ can only be considered as a weighted sum of square of effect coefficients plus the weighted sum of product of two effect coefficients. Thus by intuition, we multiply threshold Δ_0^2 and Δ_1^2 by $\sigma_{ii}^{-1} + \sum_{i=1}^{p}\sum_{j<i}2\sigma_{ij}^{-1}$ divided by the number of factors in the group p at every stage when testing the importance of the group. This is what we have for λ_0^2 and λ_1^2 in the test procedure. When there is only one factor in the group, the hypothesis we are testing is:

$$H_0 : \beta_i^2\sigma_{ii}^{-1} = \lambda_0^2 = \Delta_0^2\sigma_{ii}^{-1} \text{ against } H_1 : \beta_i^2\sigma_{ii}^{-1} = \lambda_1^2 = \Delta_1^2\sigma_{ii}^{-1}$$

This is actually testing $|\beta_i| = \Delta_0$ against $|\beta_i| = \Delta_1$.

Since some approximations have been used to modify the sequential test, we can not guarantee even stepwise power control in each splitting stage. But in the evaluation section, we can see that this screen method controls type II error strictly. On the other hand, type

I error can always be controlled during SFD-MT. Here is a short proof of it:

$$\begin{aligned} Pr(\text{Type I error}) &= Pr(\text{Classify } \beta_i \text{ as important}|\beta_i \text{ is unimportant}) \\ &= Pr(\text{Classify all groups contain } \beta_i \text{ as important}|\beta_i \text{ is unimportant}) \\ &<= Pr(\text{Classify size=1 group contains } \beta_i \text{ as important}|\beta_i \text{ is unimportant}) \\ &= \alpha \end{aligned}$$

4.2 Modified CSB

The only difference between CSB by Wan et al.[3] and modified CSB is at each splitting stage. The general frameworks of the two are exactly the same. Assume that the group being tested has p factors and all the factors have two level settings. The binary response model with logit link is written as:

$$\eta = log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

where β_0 is the intercept. We define that a model with level k as

$$\eta(k) = \beta_0 + \beta_1 + ... + \beta_k - \beta_{k+1} - ... - \beta_p$$

Thus in a model with level k, we set the first k factors in their higher levels and factors $k+1$ to p in their lower levels [3]. Notice that $\frac{1}{2}(\eta(k_1) - \eta(k_2)) = \sum_{i=k_1+1}^{k_2} \beta_i$, so if we want to test the importance of a group containing factors $k_1+1, k_1+2, ...k_2,\ k_1 < k_2$, $\widehat{\eta}(k_1)$ and $\widehat{\eta}(k_2)$ can be used to estimate the value of $\sum_{i=k_1+1}^{k_2} \beta_i$.

We generate observations in batches in the model with level k and each batch is called a replication which contains l_c observations. Define $m_k(i)$ as the number of successes in the

ith replication for level k, then we have the log odds ratio $LO_k(i)$ to be

$$LO_k(i) = log[m_k(i)/(l - m_k(i))]$$

To test the effect of the group that contains factors $k_1, k_1 + 1, ..., k_2$, the average of the difference of the log odds ratio is used. Assume that for model with level k_i, there are n_i replications already being generated in previous splitting stage, then

$$D(k_2, k_1)(j) = \frac{1}{j}\sum_{i=1}^{j}(LO_{k_2}(i) - LO_{k_1}(i)), j = min(n_{k_1}, n_{k_2})$$

is used as an estimate of the group effect. By large sample theory, if none of $m_{k_1}(i)$, $l_c - m_{k_1}(i)$, $m_{k_2}(i)$, $l_c - m_{k_2}(i)$ are very small, $LO_{k_2}(i) - LO_{k_1}(i)$ is asymptotically normally distributed with mean $2(\beta_{k_1} + \beta_{k_1+1} + ... + \beta_{k_2})$ and standard deviation $\sqrt{\frac{1}{m_{k_1}(i)} + \frac{1}{l_c - m_{k_1}(i)} + \frac{1}{m_{k_2}(i)} + \frac{1}{l_c - m_{k_2}(i)}}$ [23]. Thus, if the number of observations within each replication, l_c, is set to be large enough, we can treat $D(k_2, k_1)(j)$ as normally distributed response. The sequential test proposed by Jackson and Bradley [21] can be used here to test the hypothesis:

$$H_0 : \sum_{i=k_1}^{k_2} \beta_i = \Delta_0 \text{ against } H_1 : \sum_{i=k_1}^{k_2} \beta_i = \Delta_1$$

The framework of the modified CSB is similar to the CSB [3] and is described bellow. Some notations used are the same as in section 3.1.

Initialization: Create an empty FIFO queue. Construct a group that contains all the factor effects ($k_1 = 1, k_2 = p$) and put this group into the FIFO queue.

While FIFO queue is not empty, do

Remove a group from the FIFO queue and test the importance of the group effect. Assume that the group contains factors $k_1, k_1 + 1, ..., k_2, k_1 \leq k_2$. First check n_{k_1} and n_{k_2}, if n_{k_i} is less than the initial number of replications N_0, then generate $N_0 - n_{k_i}$ replications for model at level k_i and update the value of n_{k_i}. Then set $n = min(n_{k_1}, n_{k_2})$ and calculate the

following statistic:

$$D(k_2,k_1)(n) = \frac{1}{n}\sum_{i=1}^{n}(LO_{k_2}(i) - LO_{k_1}(i))$$

$$S(k_2,k_1)^2(n) = \frac{1}{n-1}\sum_{i=1}^{n}(LO_{k_2}(i) - LO_{k_1}(i) - D(k_2,k_1)(n))^2$$

$$T^2(n) = n * D(k_2,k_1)(n)^2/S(k_2,k_1)(n)^2$$

$$\lambda_1^2(n) = (2\triangle_1)^2/S(k_2,k_1)^2(n)$$

$$\lambda_0^2(n) = (2\triangle_0)^2/S(k_2,k_1)^2(n)$$

$$Ratio(n) = exp(-n(\lambda_1(n)^2 - \lambda_0(n)^2)/2)\frac{{}_1F_1(n/2,k/2;n\lambda_1(n)^2T^2(n)/2(n-1+T^2(n)))}{{}_1F_1(n/2,k/2;n\lambda_0(n)^2T^2(n)/2(n-1+T^2(n)))}$$

While $\gamma/(1-\alpha) \leq Ratio(n) \leq (1-\gamma)/\alpha$ **do**

Set $n = n+1$. Now if $n_{k_i}, i = 1,2$, is less than n, generate one more replication at level n_{k_i} and update the value of n_{k_i}. Update the value of $D(k_2,k_1)(n)$, $S(k_2,k_1)^2(n)$, $T^2(n)$, $\lambda_1^2(n)$, $\lambda_0^2(n)$, $Ratio(n)$.

End

If $Ratio(n) > (1-\gamma)/\alpha$ and group size equals to one, **then** classify the factor in the group as important.

If $Ratio(n) > (1-\gamma)/\alpha$ and group size is greater than one, **then** classify the group as important and split it into two subgroups. All factors in the first subgroup have smaller index than factors in the second subgroup. Put two subgroups into the FIFO queue.

If $Ratio(n) < \gamma/(1-\alpha)$, **then** classify the group as unimportant and also classify all factors in the group as unimportant.

End

4.3 Performance of Hybrid Method

The CSB can control the overall type I error and stepwise type II error [3]. But the modified CSB has implemented the asymptotically normally distribution of the log odds ratio, so the error control of modified CSB can not be guaranteed here. As indicated in section 4.14, the SFD-MT can guarantee the overall type I error but can only control stepwise type II error approximately. Thus, we can only demonstrate the performance of the hybrid method in the empirical study. Another thing needs to be mentioned here is that the factor misassignment in the prescreening stage can also affect the performance of the screening procedure. As indicated by Shen et al. [12], there are three types of factor misassignment in the prescreening stage: (1) an important factors being assigned to P-UNIMP or N-UNIMP; (2) an unimportant factors being assigned to IMP; (3) a factor with positive coefficient being assigned to N-UNIMP or a factor with negative coefficient being assigned to P-UNIMP. All the three misassignments have no effect on the overrall type I error control while the second and third misassignment would effect the overall power of the hybrid method. In the next chapter, we will see that the effect of the factor misassignments to the screening procedure is very small.

Chapter 5

Empirical Evaluation

A series of simulation-based experiments have been conducted to study the performance of the hybrid method on binary response model. The hybrid method are compared with SFD-MT in the 100-factor case experiment and also compared with modified CSB in the 200-factor case and 300-factor case. For each case, 500 independent trials are run and the average of simulation results are presented in tables. Resolution III factorial design is used in the prescreening experiment of the hybrid method and there are 30 observations at each design point in the prescreen experiment. The initial number of replications in SFD-MT is set to be equal to the number of factors in IMP plus 10 in order to make sure that the sequential test can proceed successfully. For the hybrid method, different splitting threshold Δ_t is used to find the optimum one based on efficiency. The performance of screening method is measured by two criterion: (1) the simulation runs it requires; (2) the probability it classifies factors into the correct group. In our experiments, the probability of a factors being classified as important is estimated by the percent of times it is classified as important in the 500 trials. Some parameter settings used in all the three cases are listed in the table 5.1.

5.1 100-factor case

In the 100-factor case, we generate the effect coefficients in the following way: 5% equal to $\Delta_1 = 0.2$; 5% are uniformly distributed between $\Delta_0 = 0.05$ and $\Delta_1 = 0.2$; 5% equal to $\Delta_0 = 0.05$. The sign of the coefficients are generated randomly with equal chance to

Table 5.1: Some parameter settings for the screen procedure

Parameter	Value
Δ_0	0.05
Δ_1	0.2
α	0.05
γ	0.05
N_0 in modified CSB	30
l_c	20
l	10

be positive or negative. The simulation experiment results are listed in table 5.2. The first column shows the names of the factors and the second column gives the corresponding factor effect coefficients. Here, β_1 through β_5 are important ones and we expect that they can be classified as important with a probability higher than 0.95; β_6 through β_{10} are in the indifferent zone so we do not care if they are classified as important; β_{11} through β_{15} are unimportant ones and we expect that the probability to classify them as important is less than 0.05. Columns 3 to 7 of table 5.2 shows the percent of times of the factors to be classified as important by corresponding screening procedure. The third column shows the results for SFD-MT and columns 4 to 7 are for the hybrid method with different splitting threshold. The splitting thresholds are set to be equal to Δ_0, $1.3\Delta_0$, $1.5\Delta_0$ and $1.7\Delta_0$ as shown in the table. Here, Table 5.2 only lists 15 factors since all the factors with zero effect have an estimated probability of 0 (or very close to zero) to be classified as important. The simulation results show that the SFD-MT could give a highly accurate factor screening. For example, β_1 through β_5 are all classified as important with a probability of 0.98, while unimportant factors β_{11} through β_{15} are classified as important with probability zero. However, the efficiency of SFD-MT is proved to be low since it costs nearly $150,000$ simulation runs. On the other hand, the hybrid method meets the error control requirement and uses only 25% of simulation runs as SFD-MT does. The low efficiency of SFD-MT is caused by the use of the fractional factorial design to generate replications for all factors in every splitting stage.

Table 5.2: Simulation experiment results of 100-factor case

Factor Index	Factor Effect	SFD-MT	Δ_0	$1.3 * \Delta_0$	$1.5 * \Delta_0$	$1.7 * \Delta_0$
β_1	0.200	0.98	1.00	1	1	1
β_2	-0.200	0.98	0.99	1	0.99	0.99
β_3	-0.200	0.98	1.00	0.99	1	1
β_4	-0.200	0.98	1.00	1	1	1
β_5	0.200	0.98	1.00	0.99	1	1
β_6	-0.110	0.38	0.40	0.36	0.44	0.49
β_7	0.082	0.00	0.07	0.1	0.13	0.19
β_8	0.098	0.02	0.17	0.21	0.27	0.35
β_9	-0.175	0.98	1.00	0.96	0.98	1
β_{10}	0.131	0.95	0.91	0.73	0.83	0.79
β_{11}	0.050	0.00	0.00	0.01	0.01	0.01
β_{12}	-0.050	0.00	0.00	0.01	0	0
β_{13}	-0.050	0.00	0.00	0	0	0.02
β_{14}	0.050	0.00	0.00	0.01	0	0.03
β_{15}	0.050	0.00	0.00	0	0.01	0.03
number of simulation runs		147962	35534	**33718**	35504	36825

Most of factors are unimportant and it is not worthy to waste too many observations on those factors. As the group size grows larger, this drawback of SFD-MT would be more obvious.

Table 5.3 shows the simulation runs needed in prescreen stage, stage of modified CSB and stage of SFD-MT in hybrid method. As the splitting threshold Δ_t changes from Δ_0 to $1.7\Delta_0$, we can see that when Δ_t equals to $1.3\Delta_0$, the hybrid method runs with the highest efficiency. However, we can not conclude that the optimum splitting threshold in 100-factor case is $1.3\Delta_0$, since the optimal value varies with the coefficients of factors in the group being tested.

5.2 200-factor case and 300-factor case

In the simulation experiment of the 200-factor and 300-factor cases, all the factor effects have known direction in order to meet the requirement of conducting modified CSB. We only

Table 5.3: Assignment of simulation runs for hybrid method in 100-factors case

	SFD-MT	Δ_0	$1.3 * \Delta_0$	$1.5 * \Delta_0$	$1.7 * \Delta_0$
Prescreen	0	3840	3840	3840	3840
Modified CSB	0	12875	17289	20996	24153
SFD-MT	147962	18819	12589	10667	8832
Total	147962	35534	**33718**	35504	36825

compare modified CSB and hybrid method in these two cases since SFD-MT is inefficient in large-scale problem. For the effect coefficients, 2.5% equal to $\Delta_1 = 0.2$; 5% are uniformly distributed between $\Delta_0 = 0.05$ and $\Delta_1 = 0.2$; 2.5% equal to $\Delta_0 = 0.05$. All other effect coefficient are set to zero. Modified CSB are run in two scenarios: (1) All factors are randomly distributed in the group. (2) Important factors are clustered together which is the optimal case for modified CSB. Effect coefficients are generated for 200 and 300 factors case and the simulation results for all non-zero effect coefficients are listed in Table 5.4 and Table 5.5. The column CSB(1) and CSB(2) represent the scenario one and two for modified CSB respectively. The simulation results show that the modified CSB works well when important factors are clustered. It meets the error control requirement strictly and does not require too many experiment runs. But in scenario one when factors are distributed randomly in the group, modified CSB can not meet the error control requirement and the number of runs required increase dramatically. For example, in the 200-factor case, the modified CSB uses $183,830$ runs in scenario one but only $45,205$ in scenario two. And for the important factor β_1, modified CSB classified it as important in a probability 0.922 in the scenario one, which is less than the target value 0.95; while in scenario two, the probability increases to 0.998. On the other hand, the hybrid method is stronger than modified CSB in detecting the important factors (factors with effect coefficient 0.2) in both scenarios since it has higher estimated probability in classifying them as important. And if the splitting threshold Δ_t is chosen appropriately, the number of simulation runs used is 41551 and 61045 in the 200-factor case and 300-factor case compared with 45205 and 65704 used in modified CSB in the

Table 5.4: Simulation experiment results of 200-factor case

Factor Index	Factor Effect	CSB(1)	CSB(2)	Δ_0	$1.3 * \Delta_0$	$1.5 * \Delta_0$	$1.7 * \Delta_0$
β_1	0.200	0.922	0.998	1.00	1.00	1.00	1.00
β_2	0.200	0.974	0.992	1.00	1.00	1.00	1.00
β_3	0.200	1.00	1.00	0.992	1.00	1.00	1.00
β_4	0.200	0.964	0.974	1.00	1.00	1.00	1.00
β_5	0.200	0.826	0.994	1.00	1.00	1.00	1.00
β_6	0.190	0.976	1.00	1.00	1.00	1.00	1.00
β_7	0.177	0.996	1.00	0.99	1.00	1.00	1.00
β_8	0.164	0.954	0.98	1.00	1.00	0.99	1.00
β_9	0.161	0.678	0.942	0.99	1.00	1.00	1.00
β_{10}	0.152	0.926	0.98	0.99	0.99	0.96	0.99
β_{11}	0.148	0.078	0.97	0.96	0.95	0.98	0.98
β_{12}	0.148	0.84	0.758	0.98	0.95	0.99	0.97
β_{13}	0.109	0.64	0.446	0.49	0.37	0.42	0.66
β_{14}	0.076	0.006	0.132	0.15	0.12	0.23	0.32
β_{15}	0.055	0	0.142	0	0.02	0.05	0.02
β_{16}	0.050	0.342	0.04	0	0.03	0.05	0.04
β_{17}	0.050	0.372	0.014	0.03	0.03	0.01	0.03
β_{18}	0.050	0.004	0.024	0.01	0.03	0.04	0.09
β_{19}	0.050	0.048	0.006	0.02	0.02	0.02	0.01
β_{20}	0.050	0.008	0	0.01	0.05	0.03	0.06
number of simulation runs		183830	45205	45271	**41551**	44617	45049

scenario two. This indicates that the hybrid method can save the simulation runs at about 10% and 5% in 200-factor case and 300-factor case respectively than modified CSB in its optimal scenario. Also, note that the hybrid method does not require the known direct of factor effects, which is necessary for modified CSB.

Table 5.5: Simulation experiment results of 300-factor case

Factor Index	Factor Effect	CSB(1)	CSB(2)	Δ_0	$1.3 * \Delta_0$	$1.5 * \Delta_0$	$1.7 * \Delta_0$
β_1	0.200	0.99	0.99	1.00	1.00	1.00	1.00
β_2	0.200	0.99	1.00	1.00	1.00	1.00	1.00
β_3	0.200	0.99	1.00	1.00	1.00	1.00	1.00
β_4	0.200	1.00	1.00	1.00	1.00	1.00	1.00
β_5	0.200	1.00	0.99	1.00	1.00	1.00	1.00
β_6	0.200	0.97	0.99	1.00	1.00	1.00	1.00
β_7	0.200	0.80	0.98	1.00	1.00	1.00	1.00
β_8	0.200	0.60	1.00	1.00	1.00	1.00	1.00
β_9	0.193	0.20	0.99	1.00	1.00	1.00	1.00
β_{10}	0.181	0.39	0.88	1.00	1.00	1.00	1.00
β_{11}	0.178	0.99	1.00	1.00	1.00	1.00	1.00
β_{12}	0.165	0.85	1.00	1.00	1.00	1.00	0.99
β_{13}	0.163	0.99	0.59	1.00	1.00	1.00	0.98
β_{14}	0.161	0.89	0.40	1.00	1.00	0.99	0.99
β_{15}	0.153	0.00	0.99	0.99	1.00	0.99	0.99
β_{16}	0.127	0.72	0.84	0.9	0.87	0.85	0.84
β_{17}	0.096	0.23	0.30	0.51	0.15	0.17	0.28
β_{18}	0.076	0.10	0.41	0.29	0.18	0.15	0.19
β_{19}	0.075	0.87	0.45	0.2	0.12	0.24	0.22
β_{20}	0.075	0.00	0.36	0.24	0.12	0.2	0.27
β_{21}	0.065	0.00	0.18	0.23	0.15	0.12	0.21
β_{22}	0.053	0.00	0.02	0.07	0.06	0.1	0.13
β_{23}	0.050	0.68	0.01	0.06	0.11	0.06	0.09
β_{24}	0.050	0.86	0.00	0.01	0.07	0.11	0.07
β_{25}	0.050	0.74	0.01	0.05	0.03	0.07	0.05
β_{26}	0.050	0.24	0.00	0.04	0.08	0.09	0.06
β_{27}	0.050	0.37	0.01	0.04	0.05	0.08	0.03
β_{28}	0.050	0.02	0.00	0.05	0.09	0.11	0.06
β_{29}	0.050	0.00	0.00	0.10	0.07	0.07	0.06
β_{30}	0.050	0.00	0.00	0.02	0.03	0.06	0.12
number of simulation runs		368180	65074	68649	**61045**	68140	70322

Chapter 6

Conclusion

This thesis work proposed two simulation-based factor screening methods, SFD-MT and modified CSB, which can be used on binary response model. SFD-MT is a new method combining sequential bifurcation process and multivariate sequential test. Few prior knowledge and assumptions are needed before it can be implemented. The empirical study have demonstrated the effectiveness of both SFD-MT and modified CSB. The drawback of SFD-MT is its low efficiency when used in large-scale screening problems. The modified CSB requires the known direction of factor effects and works well only when important factors are clustered, which is usually impossible in the real case. The hybrid method has been proved to be superior than SFD-MT and modified CSB in both efficiency and robustness. The pre-screening stage gives a rough estimation of the effect coefficients and allows SFD-MT and modified CSB to be implemented efficiently in the following screening process. Since the hybrid method is the combination of SFD-MT and modified CSB, there are a lot of parameters, such as the splitting threshold, the number of observations at each design point in pre-screening stage, need to be determined before conducting the screening procedure. Unfortunately, this thesis work does not provide a way to find the optimum parameter settings, which is an interesting topic for the further research.

Appendix A

Maximum Likelihood Estimation

Let the $p \times 1$ vector $\boldsymbol{\beta}$ be factor effect coefficients and the $n \times p$ matrix $\boldsymbol{X}$ be design matrix. At each design point, the number of observations is $l_i, i = 1, 2, ...n$. Then by Newton-Raphson method, the MLE of $\boldsymbol{\beta}$ in logistic regression can be calculated by the following iteration:

$$\boldsymbol{\beta^{(k+1)}} = \boldsymbol{\beta^{(k)}} + (\boldsymbol{X W^{(k)} X})^{-1} \boldsymbol{X^T} (\boldsymbol{y} - \boldsymbol{\mu^{(k)}})$$

$\boldsymbol{W^{(k)}}$ is a $n \times n$ matrix with $l_i \widehat{p}_i^{(k)} (1 - \widehat{p}_i^{(k)})$ on the diagonal and zeros everywhere else. $\boldsymbol{\mu}$ is a $n \times 1$ vector with $\boldsymbol{\mu}(i) = l_i \widehat{p}_i^{(k)}$. $\widehat{p}_i^{(k)}$ is the probability of success for response $\boldsymbol{y}(i)$ estimated at iteration k. A usual way to get initial value $\boldsymbol{\beta^{(0)}}$ is to use the least square estimator $\boldsymbol{\beta^{(0)}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T y}$ [17]. The iteration continues until the change between current estimation and the previous one is less than a user-specified accurate level.

The convergency is sometimes a problem when $\boldsymbol{X}$ is poorly designed or one of p_i is closed to one or zero [17]. When coded in Matlab, a boundary value can be assigned to the odds if the estimate odds is infinity or zero.

Appendix B

The Matlab program code for the Hybrid Method

Listing B.1: First-stage pre-screen in the Hybrid Method in 100-factor case

```
%File name Hybridsmall.m
%Input: Index of the factor in the group

function []=Hybridsmall(totalindex)
%Hybrid method screen group with unknown effect direct

global beta             %coefficient
global intercept;       %intercept coefficient
global delta0           %lower threshold
global group1           %potential important group
global error;           %error in the iterative argolrith in MLE;
global repfirststage;   %repliction at each deisgn point
global estimate         %rough estimate in the pre-screen stage
global totalsample1     %number of runs in the pre-screen stage
global totalsample2     %number of runs in modified CSB
global totalsample3     %number of runs in SFD-MT
global replication      %number of runs required at each level in modified CSB
global N0               %initial sample size in SFD-MT
global sigfactor        %important factors by screening procedure
global record2
%record of log-odds ratio at each level in modified CSB in positive group
global numofrep2
%record of number of replication at each level in modified CSB in positive group
global groupindex2
%record the index of factors used in CSB in positive group
global record3
%record of log-odds ratio at each level in modified CSB in negative group
global numofrep3
%record of number of replication at each level in modified CSB in nagative group
global groupindex3
%record the index of factors used in CSB in nagative group
global samplerecord;  %record of sample generated in SFD-MT
global samplecov;       %sample covariance matrix in SFD-MT
global B                %sample average in SFD-MT
```

```
global n;                %replication generated in SFD-MT

%initialize
totalsample1=0;
totalsample2=0;
totalsample3=0;
p=length(totalindex);
currentfactor=[intercept;beta]; % intercept plus factor effect

%MLE in pre-screen
Matrix=DesignM(100);
%generate design matrix of resolution III FFD design in the pre-screening stage
[rowdim coldim]=size(Matrix);
D=[ones(rowdim,1) Matrix];
[rowdim colddim]=size(D);
Y=zeros(rowdim,1);
originpi=zeros(rowdim,1);
for i = 1:rowdim
    exp(D(i,:)*currentfactor)/(1+exp(D(i,:)*currentfactor));
    Y(i,1)=binornd(repfirststage, exp(D(i,:)*currentfactor)/...
    (1+exp(D(i,:)*currentfactor)));
    originpi(i,1)=exp(D(i,:)*currentfactor)/(1+exp(D(i,:)*currentfactor));%%%%
end
M=ones(rowdim,1)*repfirststage;
response=zeros(rowdim,1);
temp=0;
for i = 1: rowdim
    temp=log(Y(i,1)/(repfirststage-Y(i,1)));
    if temp > 3
        temp= 3;
    else if temp < -3
            temp = -3;
        end
    end
    response(i,1)=temp;
end
beta0= inv(D'*D)*D'*response;%starting value by linear regression
old=beta0;
new=0;
tempt=beta0;
iter=0;
pi=zeros(rowdim,1);
Wvector=zeros(rowdim,1);
mu=zeros(rowdim,1);
while (norm(tempt-new))^0.5 > error
    tempt=old;
    for i=1:rowdim
        pi(i,1)= exp(D(i,:)*old)/(1+exp(D(i,:)*old));
        Wvector(i)=M(i,1)*pi(i,1)*(1-pi(i,1));
```

```
        mu(i,1)=M(i,1)*pi(i,1);
    end
    W=diag(Wvector);
    new=old+inv(D'*W*D)*D'*(Y-mu);
    old=new;
    iter=iter+1;
    %iter
end
estimate=new;
totalsample1=totalsample1+rowdim*repfirststage;

%partition factor in potential important group and unimportant group
group1=[]; % potential important group
group2=[]; % potential unimportant positve group
group3=[]; % potential unimportant nagative group
for i=2:p+1
    if abs(estimate(i))>=1.9*delta0;
        group1=[group1 i-1];
    end
    if estimate(i) < 1.9*delta0 && estimate(i)>0
        group2=[group2 i-1];
    end
    if estimate(i) <=0 && estimate(i)>(-1.9*delta0)
        group3=[group3 i-1];
    end
end
value2=zeros(1,length(group2));
value3=zeros(1,length(group3));
for j=1:length(group2)
    value2(j)=estimate(group2(j)+1);
end
for j=1:length(group3)
    value3(j)=estimate(group3(j)+1);
end
group2=[group2;value2];
group3=[group3;value3];
group2=sortrows(group2',2);
group3=sortrows(group3',2);
a2=zeros(length(group2),1);
a3=zeros(length(group3),1);
for i=1:length(group2)
    a2(i,1)=i;
end
for i=1:length(group3)
    a3(i,1)=i;
end
groupindex2=[group2(:,1) a2]';
groupindex3=[group3(:,1) a3]';
record2=zeros(length(group2)+1,2000);
```

```
record3=zeros(length(group3)+1,2000);
numofrep2=zeros(length(group2)+1,1);
numofrep3=zeros(length(group3)+1,1);

%CSB screen unimportant positive group
csb2(groupindex2);
totalsample2=totalsample2+sum(numofrep2)*replication;
%CSB screen unimportant nagative group
csb3(groupindex3);
totalsample2=totalsample2+sum(numofrep3)*replication;

%SFD-MT
N0=length(group1)+10;
logitscreen;
```

Listing B.2: First-stage pre-screen in the Hybrid Method in 200-factor case

```
%File name: Hybridslarge.m
%Input: total index of factors in the group

function []=Hybridlarge(totalindex)
%Hybride method screen factors with known effect direct

global beta            %coefficient
global intercept;      %intercept coefficient
global delta0          %lower threshold
global group1          %potential important group
global error;          %error in the iterative argolrith in MLE;
global repfirststage; %repliction at each deisgn point
global estimate        %rough estimate in the pre-screen stage
global totalsample1    %number of runs in the pre-screen stage
global totalsample2    %number of runs in modified CSB
global totalsample3    %number of runs in SFD-MT
global replication     %number of runs required at each level in modified CSB
global N0              %initial sample size in SFD-MT
global sigfactor       %important factors by screening procedure
global record2
%record of log-odds ratio at each level in modified CSB in positive group
global numofrep2
%record of number of replication at each level in modified CSB in positive group
global groupindex2
%record the index of factors in original factor group and new group when used in CSB
global samplerecord;  %record of sample generated in SFD-MT
global samplecov;      %sample covariance matrix in SFD-MT
global B               %sample average in SFD-MT
global n;              %replication generated in SFD-MT

%initializing
totalsample1=0;
totalsample2=0;
totalsample3=0;
sigfactor=[];
p=length(totalindex);
currentfactor=[intercept;beta]; % add intercept terms into the factor vector

%MLE in pre-screen
Matrix=DesignM(200);
%generate design matrix of resolution III FFD in the pre-screening stage
[rowdim coldim]=size(Matrix);
D=[ones(rowdim,1) Matrix];
[rowdim colddim]=size(D);
Y=zeros(rowdim,1);
originpi=zeros(rowdim,1);
for i = 1:rowdim
```

```
    exp(D(i,:)*currentfactor)/(1+exp(D(i,:)*currentfactor));
    Y(i,1)=binornd(repfirststage, exp(D(i,:)*currentfactor)/...
    (1+exp(D(i,:)*currentfactor)));
    originpi(i,1)=exp(D(i,:)*currentfactor)/(1+exp(D(i,:)*currentfactor));%%%%
end
M=ones(rowdim,1)*repfirststage;

response=zeros(rowdim,1);
temp=0;
for i = 1: rowdim
    temp=log(Y(i,1)/(repfirststage-Y(i,1)));
    if temp > 3
        temp= 3;
    else if temp < -3
            temp = -3;
        end
    end
    response(i,1)=temp;
end
beta0= inv(D'*D)*D'*response; %starting value by linear regression

old=beta0;
new=zeros(p+1,1);
tempt=beta0;
estimate=zeros(p+1,1);
iter=0;
pi=zeros(rowdim,1);
Wvector=zeros(rowdim,1);
mu=zeros(rowdim,1);
while (norm(tempt-new))^0.5 > error
    tempt=old;
    for i=1:rowdim
        pi(i,1)= exp(D(i,:)*old)/(1+exp(D(i,:)*old));
        Wvector(i)=M(i,1)*pi(i,1)*(1-pi(i,1));
        mu(i,1)=M(i,1)*pi(i,1);
    end
    W=diag(Wvector);
    new=old+inv(D'*W*D)*D'*(Y-mu);
    old=new;
    iter=iter+1;
end
estimate=new;
totalsample1=totalsample1+rowdim*repfirststage;

%partition factor in potential important group and unimportant group
group1=[]; %potential important group
group2=[]; %potential unimportant group
for i=2:p+1
    if abs(estimate(i))>=delta0;
```

```
        group1=[group1 i-1];
    end
    if abs(estimate(i)) < delta0
        group2=[group2 i-1];
    end
end
value2=zeros(1,length(group2));
for j=1:length(group2)
    value2(j)=estimate(group2(j)+1);
end
group2=[group2;value2];
group2=sortrows(group2',2);
a2=zeros(length(group2),1);
for i=1:length(group2)
    a2(i,1)=i;
end
groupindex2=[group2(:,1) a2]';
record2=zeros(length(group2)+1,2000);     % 2000 may not be enough
numofrep2=zeros(length(group2)+1,1);

%Modified CSB on potential unimportant group
csb2(groupindex2);
totalsample2=totalsample2+sum(numofrep2)*replication;
%SFD-MT in potential important group
N0=length(group1)+10;
logitscreen;
```

Listing B.3: Modified CSB for positive potential important group

```
%File name: csb2.m
%Input: index of factors in the positive potential important group

function []=csb2(currentindex)
%modified csb used in positive potential unimportant group

global beta
global intercept
global alpha        %type I error
global gamma        %type II error
global replication
global sbn0         % intial number of samples in sequential test used in CSB
global delta0
global delta1       %higher threshold
global kk           %times of minification
global multiple     %minifaction when calculating hypergeometrix function
global sigfactor
global record2
global numofrep2
global groupindex2

%initialize
multiple=1e-50;
kk=0;
dim=size(groupindex2);
p=dim(1,2);
currentdim=size(currentindex);
currentp=currentdim(1,2);
level2=currentindex(2,currentp);
level1=currentindex(2,1)-1;
eta1=intercept;
eta2=intercept;

%modified CSB
for i=1:level2
        eta2=eta2+beta(groupindex2(1,i));
end
for i=level2+1:p
        eta2=eta2-beta(groupindex2(1,i));
end
for i=1:level1
        eta1=eta1+beta(groupindex2(1,i));
end
for i=level1+1:p
        eta1=eta1-beta(groupindex2(1,i));
end
```

```
p1=1/(1+exp(-eta1));
p2=1/(1+exp(-eta2));
count=1;
%generate observations at level1 and level2
while numofrep2(level2+1,1)<sbn0
    m2=binornd(replication,p2);
    record2(level2+1,count)=m2;
    numofrep2(level2+1,1)=numofrep2(level2+1,1)+1;
    count=count+1;
end
count=1;
while numofrep2(level1+1,1)<sbn0
    m1=binornd(replication,p1);
    record2(level1+1,count)=m1;
    numofrep2(level1+1,1)=numofrep2(level1+1,1)+1;
    count=count+1;
end

%ratio sequential test begin
lor=0;
B=0;
samplecov=0;
samplerecord=[];
n=0;
%calculating ratio
for i=1:sbn0
   n=i;
   summation=0;
   m1=record2(level1+1,i);
   m2=record2(level2+1,i);
   %some approximation used when m2 or m1 equals to replication
   if m2==replication && m1==replication
       lor=0;
   else if m2==replication && m1~=replication && m1~=0
        lor=log((replication-0.5)*(replication-m1)/m1/0.5);
       else if m2~=replication && m2~=0 && m1==replication
               lor=log(m2*0.5/(replication-0.5)/(replication-m2));
           else if m2==0 && m1==0
                   lor=0;
               else if m2==0 && m1~=0 && m1~=replication
                       lor=log(0.5*(replication-m1)/m1/(replication-0.5));
                   else if m2~=0 && m2~=replication && m1==0
                           lor=(m2*(replication-0.5)/0.5/(replication-m2));
                       else if m2==replication && m1==0
                               lor=((replication-0.5)*(replication-0.5)/0.5/0.5);
                           else if m2==0 && m1==replication
                                   lor=(0.25/(replication-0.5)/(replication-0.5));
                               else
```

```
                                        lor=log(m2*(replication-m1)/m1/(replication-m2));
                                         % log odds ratio, asymtotic normal
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
        samplerecord=[samplerecord lor];
        B=((n-1)*B+lor)/n;
        for j=1:n
            summation=summation+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
        end
        samplecov=summation/(n-1);
    end
    Tn2=n*(B)'*inv(samplecov)*(B); %test statistics
    log1=log(confluenthyper(n/2,1/2,n*4*delta1^2*Tn2/2/(n-1+Tn2)));
    log2=log(confluenthyper(n/2,1/2,n*4*delta0^2*Tn2/2/(n-1+Tn2)));
    logratio=-4*n*(delta1^2-delta0^2)/2+log1-log2;

    while  logratio > log(gamma/(1-alpha)) && logratio < log((1-gamma)/alpha)
        %one replication added each times
        n=n+1;
        if numofrep2(level2+1,1) < n
            record2(level2+1,n)=binornd(replication,p2);
            numofrep2(level2+1,1)=numofrep2(level2+1,1)+1;
        end
        if numofrep2(level1+1,1) < n
            record2(level1+1,n)=binornd(replication,p1);
            numofrep2(level1+1,1)=numofrep2(level1+1,1)+1;
        end
        m2=record2(level2+1,n);
        m1=record2(level1+1,n);
        summation=0;
        if m2==replication && m1==replication
            lor=0;
        else if m2==replication && m1~=replication && m1~=0
            lor=log((replication-0.5)*(replication-m1)/m1/0.5);
            else if m2~=replication && m2~=0 && m1==replication
                lor=log(m2*0.5/(replication-0.5)/(replication-m2));
                else if m2==0 && m1==0
                    lor=0;
                    else if m2==0 && m1~=0 && m1~=replication
                        lor=log(0.5*(replication-m1)/m1/(replication-0.5));
                        else if m2~=0 && m2~=replication && m1==0
                            lor=(m2*(replication-0.5)/0.5/(replication-m2));
                            else if m2==replication && m1==0
```

```
                                lor=((replication-0.5)*(replication-0.5)/0.5/0.5);
                            else if m2==0 && m1==replication
                                lor=(0.25/(replication-0.5)/(replication-0.5));
                            else
                                lor=log(m2*(replication-m1)/m1/(replication-m2));
                                % log odds ratio, asymtotic normal
                            end
                        end
                    end
                end
            end
        end
    end
     samplerecord = [samplerecord lor];
     B=((n-1)*B+lor)/n;
     for j=1:n
        summation=summation+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
     end
     samplecov=summation/(n-1);
     Tn2=n*(B)'*inv(samplecov)*(B);
     log1=log(confluenthyper(n/2,1/2,n*4*delta1^2*Tn2/2/(n-1+Tn2)));
     log2=log(confluenthyper(n/2,1/2,n*4*delta0^2*Tn2/2/(n-1+Tn2)));
     logratio=-4*n*(delta1^2-delta0^2)/2+log1-log2;
end
%ratio test end%

if logratio <= log(gamma/(1-alpha))
    accept = 1;
%splitting when group size is greater than 1
else if logratio >= log((1-gamma)/alpha) && currentp>1
        accept=0;
        index1=currentindex(:,1:floor(currentp/2));
        index2=currentindex(:,floor(currentp/2)+1:currentp);
        csb2(index1)
        csb2(index2)
    %classified important factors if group size equals to 1
    else if logratio >= log((1-gamma)/alpha) && currentp==1
            accept=0;
            sigfactor=[sigfactor currentindex(1,1)];
        end
    end
end
```

Listing B.4: Modified CSB for negative potential important group

```
%File name: csb3.m
%Input: index of factors in the negative potential important group

function []=csb3(currentindex)
%modified csb used in nagative potential unimportant group
global beta
global intercept
global alpha          %type I error
global gamma          %type II error
global replication    % better > 20
global sbn0           % intial number of samples in sequential test used in CSB
global delta0
global delta1         %higher threshold
global kk             %times of minification
global multiple       %minifaction when calculating hypergeometrix function
global sigfactor
global record3
global numofrep3
global groupindex3

%initialize
multiple=1e-50;
kk=0;
dim=size(groupindex3);
p=dim(1,2);
currentdim=size(currentindex);
currentp=currentdim(1,2);
level2=currentindex(2,currentp);
level1=currentindex(2,1)-1;
eta1=intercept;
eta2=intercept;

%modified CSB
for i=1:level2
        eta2=eta2-beta(groupindex3(1,i));
end
for i=level2+1:p
        eta2=eta2+beta(groupindex3(1,i));
end
for i=1:level1
        eta1=eta1-beta(groupindex3(1,i));
end
for i=level1+1:p
        eta1=eta1+beta(groupindex3(1,i));
end
p1=1/(1+exp(-eta1));
p2=1/(1+exp(-eta2));
```

```
count=1;
%generate observations at level1 and level2
while numofrep3(level2+1,1)<sbn0
    m2=binornd(replication,p2);
    record3(level2+1,count)=m2;
    numofrep3(level2+1,1)=numofrep3(level2+1,1)+1;
    count=count+1;
end
count=1;
while numofrep3(level1+1,1)<sbn0
    m1=binornd(replication,p1);
    record3(level1+1,count)=m1;
    numofrep3(level1+1,1)=numofrep3(level1+1,1)+1;
    count=count+1;
end

%ratio sequential test begin
lor=0;
B=0;
samplecov=0;
samplerecord=[];
n=0;
for i=1:sbn0
   n=i;
   summation=0;
   m1=record3(level1+1,i);
   m2=record3(level2+1,i);
   %some approximation used when m2 or m1 equals to replication
   if m2==replication && m1==replication
       lor=0;
   else if m2==replication && m1~=replication && m1~=0
        lor=log((replication-0.5)*(replication-m1)/m1/0.5);
       else if m2~=replication && m2~=0 && m1==replication
               lor=log(m2*0.5/(replication-0.5)/(replication-m2));
           else if m2==0 && m1==0
                   lor=0;
               else if m2==0 && m1~=0 && m1~=replication
                       lor=log(0.5*(replication-m1)/m1/(replication-0.5));
                   else if m2~=0 && m2~=replication && m1==0
                           lor=(m2*(replication-0.5)/0.5/(replication-m2));
                       else if m2==replication && m1==0
                               lor=((replication-0.5)*(replication-0.5)/0.5/0.5);
                           else if m2==0 && m1==replication
                                   lor=(0.25/(replication-0.5)/(replication-0.5));
                               else
                                   lor=log(m2*(replication-m1)/m1/(replication-m2));
                                   % log odds ratio, asymtotic normal
                               end
                           end
```

```
                    end
                end
            end
        end
    end
    samplerecord=[samplerecord lor];
    B=((n-1)*B+lor)/n;
    for j=1:n
        summation=summation+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
    end
    samplecov=summation/(n-1);
end
Tn2=n*(B)'*inv(samplecov)*(B); %test statistics
log1=log(confluenthyper(n/2,1/2,n*4*delta1^2*Tn2/2/(n-1+Tn2)));
log2=log(confluenthyper(n/2,1/2,n*4*delta0^2*Tn2/2/(n-1+Tn2)));
logratio=-4*n*(delta1^2-delta0^2)/2+log1-log2;

while  logratio > log(gamma/(1-alpha)) && logratio < log((1-gamma)/alpha)
    n=n+1;
    if numofrep3(level2+1,1) < n
        record3(level2+1,n)=binornd(replication,p2);
        numofrep3(level2+1,1)=numofrep3(level2+1,1)+1;
    end
    if numofrep3(level1+1,1) < n
        record3(level1+1,n)=binornd(replication,p1);
        numofrep3(level1+1,1)=numofrep3(level1+1,1)+1;
    end
    m2=record3(level2+1,n);
    m1=record3(level1+1,n);
    summation=0;
   if m2==replication && m1==replication
       lor=0;
   else if m2==replication && m1~=replication && m1~=0
        lor=log((replication-0.5)*(replication-m1)/m1/0.5);
       else if m2~=replication && m2~=0 && m1==replication
               lor=log(m2*0.5/(replication-0.5)/(replication-m2));
           else if m2==0 && m1==0
                   lor=0;
               else if m2==0 && m1~=0 && m1~=replication
                       lor=log(0.5*(replication-m1)/m1/(replication-0.5));
                   else if m2~=0 && m2~=replication && m1==0
                           lor=(m2*(replication-0.5)/0.5/(replication-m2));
                       else if m2==replication && m1==0
                               lor=((replication-0.5)*(replication-0.5)/0.5/0.5);
                           else if m2==0 && m1==replication
                                   lor=(0.25/(replication-0.5)/(replication-0.5));
                               else
                                   lor=log(m2*(replication-m1)/m1/(replication-m2));
```

```
                                        % log odds ratio, asymtotic normal
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
        samplerecord = [samplerecord lor];
        B=((n-1)*B+lor)/n;
        for j=1:n
            summation=summation+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
        end
        samplecov=summation/(n-1);
        Tn2=n*(B)'*inv(samplecov)*(B);
        log1=log(confluenthyper(n/2,1/2,n*4*delta1^2*Tn2/2/(n-1+Tn2)));
        log2=log(confluenthyper(n/2,1/2,n*4*delta0^2*Tn2/2/(n-1+Tn2)));
        logratio=-4*n*(delta1^2-delta0^2)/2+log1-log2;
    end
    %ratio test end%

    if logratio <= log(gamma/(1-alpha))
        accept = 1;
    %splitting when group size is greater than 1
    else if logratio >= log((1-gamma)/alpha) && currentp>1
            accept=0;
            index1=currentindex(:,1:floor(currentp/2));
            index2=currentindex(:,floor(currentp/2)+1:currentp);
            csb3(index1);
            csb3(index2);
            %classified important factors if group size equals to 1
        else if logratio >= log((1-gamma)/alpha) && currentp==1
                accept=0;
                sigfactor=[sigfactor currentindex(1,1)];
            end
        end
    end
```

Listing B.5: SFD-MT on the entire potential important group

```
%File name: logitscreen.m
%Input: index of factors in the positive potential important group

function []= logitscreen %SFD-MT begore any splitting of the group

global sigfactor;
global gamma;    %type II error;
global alpha;    %type I error;
global N0;       %initial # of replication
global delta0;   %lower threshold;
global delta1;   %higher threshold;
global multiple;
global kk
global samplerecord;
global samplecov;
global B
global n;
global group1;

%initialize
samplerecord=[];
multiple=1e50;
kk=0;
p = length(group1); % length of factor vector include intercept
currentindex=group1;
index1=[];
index2=[];
indexofindex1=[];
indexofindex2=[];
n=0;

B =logitreg(currentindex);  %p+1 by 1 since intercept effect included
samplecov=0;
samplerecord=B;  % save the sample generated

%generate initial  number of sample, sequential test begin
for i=2:N0
    n=i;
    sum=0;
    tempt=logitreg(currentindex);
    samplerecord=[samplerecord tempt];
    B=((n-1)*B+tempt)/n;
    for j=1:n
        sum=sum+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
    end
    samplecov=sum/(n-1);
end
```

```
%calculate the threshold used in each splitting stage varing with n
invsamplecov=inv(samplecov);
sum1=0;
sum2=0;
for i=1:p
    sum1=sum1+invsamplecov(i,i);
end
for i=1:p
    for j=1:i-1
        sum2=sum2+2*invsamplecov(i,j);
    end
end
temptdelta1=delta1*((sum1+sum2)/p)^0.5;
temptdelta0=delta0*((sum1+sum2)/p)^0.5;
Tn2=n*(B)'*inv(samplecov)*(B);

%calculate log ratio, some approximation used by minification
log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
while log2 ==inf
    kk=kk+1;
    log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
end
kk2=kk;
log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
while log1 == inf
    kk=kk+1;
    log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
end
log2=log2-(kk-kk2)*log(multiple);
logratio=-n*(temptdelta1^2-temptdelta0^2)/2+log1-log2;

%No conclusion with initial number of sample, generate more replication
while  logratio > log(gamma/(1-alpha)) && logratio < log((1-gamma)/alpha)
    n=n+1; %one replication generated
    tempt=logitreg(currentindex);
    samplerecord = [samplerecord tempt];
    B=((n-1)*B+tempt)/n;
    sum=0;
    for j=1:n
       sum=sum+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
    end
    samplecov=sum/(n-1);

    %calculate the threshold used after n updated
    invsamplecov=inv(samplecov);
    sum1=0;
    sum2=0;
```

```
    for i=1:p
        sum1=sum1+invsamplecov(i,i);
    end
    for i=1:p
        for j=1:i-1
            sum2=sum2+2*invsamplecov(i,j);
        end
    end
    temptdelta1=delta1*((sum1+sum2)/p)^0.5;
    temptdelta0=delta0*((sum1+sum2)/p)^0.5;
    Tn2=n*(B)'*inv(samplecov)*(B);
    %calculate log ratio, some approximation used by minification
    log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
    log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
    while log2 ==inf
        kk=kk+1;
        log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
    end
    kk2=kk;
    log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
    while log1 == inf
        kk=kk+1;
        log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
    end
    log2=log2-(kk-kk2)*log(multiple);
    logratio=-n*(temptdelta1^2-temptdelta0^2)/2+log1-log2;
end
%sequential test end

temptsamplecov=samplecov;
if logratio <= log(gamma/(1-alpha))
    accept = 1;

else if logratio >= log((1-gamma)/alpha) && p > 2
    %spliting important group when group size is greater than 2
    accept = 0;
    max=0;
    maxindex=[];
    %splitting process
    while( length( index1 ) < p/2 )
        for i = 1:p
            for j = i+1 : p
                if abs(temptsamplecov(i,j)) >= max
                    i0=i;
                    j0=j;
                     max = abs(temptsamplecov(i0,j0));
                    maxindex=[i0 j0];
                end
            end
```

```
    end
    max=0;
    temptsamplecov(i0,j0)=0;
    already in = 0; %judge if some factor is already in the group
    for l =1:length(index1)
        if i0 == indexofindex1(l)
            already in = 1;
            break
        end
    end
    if already in == 0
        index1=[index1 currentindex(i0)];
        indexofindex1=[indexofindex1 i0];
    end
    already in = 0;
    for l =1:length(index1)
        if j0 == indexofindex1(l)
            already in = 1;
            break
        end
    end
    if already in == 0
        index1=[index1 currentindex(j0)];
        indexofindex1=[indexofindex1 j0];
    end
end
already in = 0;
for l = 1:length(currentindex)
    already in=0;
    for t=1:length(index1)
        if l == indexofindex1(t)
            already in = 1;
            break
        end
    end
   if already in == 0
       index2 = [index2 currentindex(l)];
   end
end
logitsplit(index1);  %subgroup being screened
logitsplit(index2);  %subgroup being screened
else if logratio >= log((1-gamma)/alpha) && p == 1
    %important group with size 1
    accept=0;
    sigfactor=[sigfactor currentindex];
    else if logratio >= log((1-gamma)/alpha) && p == 2
        %splitting with size two group
        index1=currentindex(1);
        index2=currentindex(2);
```

```
            logitsplit(index1);
            logitsplit(index2);
        end
    end
    end
end
```

Listing B.6: SFD-MT on the subgroups of the potential important group

```
%File name: logitsplit.m
%Input: index of factors in the subgroup of the positive potential important group

function []= logitsplit (indexvector)
%SFD-MT after splitting of the group

global sigfactor;
global gamma;      % type 2 error;
global alpha;      % type 1 error;
global delta0;     % lower threshold;
global delta1;     % higher threshold;
global multiple;
global kk;
global samplerecord;
global samplecov;
global B;
global group1;
global n;

%initialize
multiple=1e50;
kk=0;
p = length(indexvector);
currentindex=indexvector;
index1=[];
index2=[];
indexofindex1=[];
indexofindex2=[];
temptsamplecov=zeros(p,p);  %covariance matrix used when testing subgroup
temptB=zeros(p,1);
for i=1:p
    for j=1:p
        temptsamplecov(i,j)=samplecov...
        (length(group1(group1<=currentindex(i))),...
        length(group1(group1<=currentindex(j))));
    end
end
for i=1:p
    temptB(i,1)=B(length(group1(group1<=currentindex(i))));
end

%calculate the threshold used in each splitting stage varing with n
invsamplecov=inv(temptsamplecov);
sum1=0;
sum2=0;
for i=1:p
     sum1=sum1+invsamplecov(i,i);
```

```
end
for i=1:p
    for j=1:i-1
        sum2=sum2+2*invsamplecov(i,j);
    end
end
temptdelta1=delta1*((sum1+sum2)/p)^0.5;
temptdelta0=delta0*((sum1+sum2)/p)^0.5;

%%calculate log ratio, some approximation used by minification
Tn2=n*(temptB)'*inv(temptsamplecov)*(temptB);
log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
while log2 ==inf
    kk=kk+1;
    log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
end
kk2=kk;
log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));

while log1 == inf
    kk=kk+1;
    log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
end
log2=log2-(kk-kk2)*log(multiple);
logratio=-n*(temptdelta1^2-temptdelta0^2)/2+log1-log2;

%No conlustion got with current sample size, generate more replicaiton
while  logratio > log(gamma/(1-alpha)) && logratio < log((1-gamma)/alpha)
    n=n+1  ;   %one replication generated
    tempt=logitreg(group1);
    samplerecord = [samplerecord tempt];
    B=((n-1)*B+tempt)/n;
    sum=0;
    for j=1:n
       sum=sum+((samplerecord(:,j)-B)*(samplerecord(:,j)-B)');
    end
    samplecov=sum/(n-1);
    for i=1:p
        for j=1:p
            temptsamplecov(i,j)=samplecov...
            (length(group1(group1<=currentindex(i))),...
            length(group1(group1<=currentindex(j))));
        end
    end
    for i=1:p
        temptB(i,1)=B(length(group1(group1<=currentindex(i))));
    end
```

```
    %calculate the threshold used in each splitting stage varing with n
    invsamplecov=inv(temptsamplecov);
    sum1=0;
    sum2=0;
    for i=1:p
        sum1=sum1+invsamplecov(i,i);
    end
    for i=1:p
        for j=1:i-1
            sum2=sum2+2*invsamplecov(i,j);
        end
    end
    temptdelta1=delta1*((sum1+sum2)/p)^0.5;
    temptdelta0=delta0*((sum1+sum2)/p)^0.5;
    Tn2=n*(temptB)'*inv(temptsamplecov)*(temptB);

    %calculate log ratio, some approximation used by minification
    log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
    log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
    while log2 ==inf
        kk=kk+1;
        log2=log(confluenthyper(n/2,p/2,n*temptdelta0^2*Tn2/2/(n-1+Tn2)));
    end
    kk2=kk;
    log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
    while log1 == inf
        kk=kk+1;
        log1=log(confluenthyper(n/2,p/2,n*temptdelta1^2*Tn2/2/(n-1+Tn2)));
    end
    log2=log2-(kk-kk2)*log(multiple);
    logratio=-n*(temptdelta1^2-temptdelta0^2)/2+log1-log2;
end
%sequential test end

if logratio <= log(gamma/(1-alpha))
    accept = 1;
else if logratio >= log((1-gamma)/alpha) && p > 2
    %spliting important group when group size is greater than 2
    accept = 0;
    max=0;
    maxindex=[];
    while( length( index1 ) < p/2 )
        for i = 1:p
            for j = i+1 : p
                if abs(temptsamplecov(i,j)) >= max
                    i0=i;
                    j0=j;
                     max = abs(temptsamplecov(i0,j0));
                    maxindex=[i0 j0];
```

```
            end
        end
    end
    max=0;
    temptsamplecov(i0,j0)=0;
    already in = 0; %judge if some factor is already in the group
    for l =1:length(index1)
        if i0 == indexofindex1(l)
            already in = 1;
            break
        end
    end
    if already in == 0
        index1=[index1 currentindex(i0)];
        indexofindex1=[indexofindex1 i0];
    end

    already in = 0;
    for l =1:length(index1)
        if j0 == indexofindex1(l)
            already in = 1;
            break
        end
    end
    if already in == 0
        index1=[index1 currentindex(j0)];
        indexofindex1=[indexofindex1 j0];
    end
end

already in = 0;

for l = 1:length(currentindex)
    already in=0;
    for t=1:length(index1)
        if l == indexofindex1(t)
            already in = 1;
            break
        end
    end
   if already in == 0
       index2 = [index2 currentindex(l)];
   end

end
logitsplit(index1);  %subgroup being screened
logitsplit(index2);  %subgroup being screened

else if logratio >= log((1-gamma)/alpha) && p == 1
```

```
        %important group with size 1
        accept=0;
        sigfactor=[sigfactor currentindex];

    else if logratio >= log((1-gamma)/alpha) && p == 2
            %splitting with size two group
            accept=0;
            index1=currentindex(1);
            index2=currentindex(2);
            logitsplit(index1);
            logitsplit(index2);
        end
    end
    end
end
```

Listing B.7: logistic regression process

```
%File name: logitreg.m
%Input: index of factors in group being tested

function [estimate]= logitreg ( currentindex )
%estimation the effect coefficient by logistic regression

global beta;
global intercept
global error; % error in MLE;
global rep;   %repliction at each deisgn point in SFD-MT
global totalsample3;

%Iinitialize
p=length(currentindex);   % intercept effect not included
currentfactor=zeros(p+1,1);
currentfactor(1,1)=intercept;
for i =2:p+1
    currentfactor(i,1)=beta(currentindex(1,i-1));
end
Matrix=DesignM(p);
[rowdim coldim]=size(Matrix);
D=[ones(rowdim,1) Matrix];
% D is the Design Matrix with intercept in the column one
coldim=coldim+1;

%MLE begins
Y=zeros(rowdim,1);
originpi=zeros(rowdim,1);
for i = 1:rowdim
   exp(D(i,:)*currentfactor)/(1+exp(D(i,:)*currentfactor));
   Y(i,1)= binornd(rep, exp(D(i,:)*currentfactor)/...
   (1+exp(D(i,:)*currentfactor)));
   originpi(i,1)=exp(D(i,:)*currentfactor)/(1+exp(D(i,:)*currentfactor));%%%%
end
M=ones(rowdim,1)*rep;
response=zeros(rowdim,1);
temp=0;
for i = 1: rowdim
    temp=log(Y(i,1)/(rep-Y(i,1)));
    if temp > 4
        temp= 4;
    else if temp < -4
            temp = -4;
        end
    end
    response(i,1)=temp;
end
```

```
beta0= inv(D'*D)*D'*response;   %initial estimation
old=beta0;
new=0;
tempt=beta0;
iter=0;
pi=zeros(rowdim,1);
Wvector=zeros(rowdim,1);
mu=zeros(rowdim,1);

while (norm(tempt-new))^0.5 > error && iter<15
     tempt=old;
     for i=1:rowdim
          pi(i,1)= exp(D(i,:)*old)/(1+exp(D(i,:)*old));
          Wvector(i)=M(i,1)*pi(i,1)*(1-pi(i,1));
          mu(i,1)=M(i,1)*pi(i,1);
     end
     W=diag(Wvector);
     new=old+inv(D'*W*D)*D'*(Y-mu);
     old=new;
     iter=iter+1;
end
estimate=zeros(p,1);
for i=1:p
     estimate(i,1)=new(i+1,1);
end
totalsample3=totalsample3+rep*rowdim; %number of runs increases
```

Listing B.8: Genarate design matrix of resolution III fraction factorial design

```
%File name: DesignM.m
%Input: number of factors in the group

function [D]=DesignM(numberofdesign)
%generate design matrix of resolution III fractional factorial design
%with up to 500 factors
M=numberofdesign; % Total number of factors in the group
% find m
if M<3
    D=(ff2n(M)-0.5)*2;
end
if M>=3
m=3;
while 2^m <= M+1
    m=m+1;
end

full=(ff2n(m)-0.5)*2 ;   % full factorial design
dimfull=size(full);
D=[full zeros(dimfull(1),M-m)];
dimension=size(D);

% now generate the m+1 to M Column
n=0;
%confounded with two-factor interaction terms
for l=1:m-1
    for k=l+1:m
        %calculate the m+n th coluumn
        n=n+1;
        if (m+n)<=dimension(2)
            for i=1:dimension(1)
                D(i,m+n)=D(i,l)*D(i,k);
            end
        end
    end
end

%confounded with three-factor interaction terms
if m+n<dimension(2)
    for l=1:m-2
        for k=l+1:m-1
            for p=k+1:m
                n=n+1;
                if m+n<=dimension(2)
                    for i=1:dimension(1)
                        D(i,m+n)=D(i,l)*D(i,k)*D(i,p);
                    end
```

```
                end
            end
        end
    end
end
%confounded with four-factor interaction terms
if m+n<dimension(2)
    for l=1:m-3
        for k=l+1:m-2
            for p=k+1:m-1
                for q=p+1:m
                n=n+1;
                if m+n<=dimension(2)
                    for i=1:dimension(1)
                        D(i,m+n)=D(i,l)*D(i,k)*D(i,p)*D(i,q);

                    end
                end
            end
        end
    end
end
%confounded with five-factor interaction terms
if m+n<dimension(2)
    for l=1:m-4
        for k=l+1:m-3
            for p=k+1:m-2
                for q=p+1:m-1
                    for o=q+1:m
                    n=n+1;
                    if m+n<=dimension(2)
                        for i=1:dimension(1)
                            D(i,m+n)=D(i,l)*D(i,k)*D(i,p)*D(i,q)*D(i,o);
                        end
                    end
                end
            end
        end
    end
end
%confounded with six-factor interaction terms
  if m+n<dimension(2)
    for l=1:m-5
        for k=l+1:m-4
            for p=k+1:m-3
                for q=p+1:m-2
                    for o=q+1:m-1
```

```
                        for h=o+1:m
                          n=n+1;
                          if m+n<=dimension(2)
                              for i=1:dimension(1)
                                  D(i,m+n)=D(i,l)*D(i,k)*D(i,p)*D(i,q)*D(i,o)*D(i,h);
                              end
                          end
                        end
                    end
                end
            end
        end
    end
end
 %confounded with seven-factor interaction terms
 if m+n<dimension(2)
    for l=1:m-5
        for k=l+1:m-4
            for p=k+1:m-3
                for q=p+1:m-2
                    for o=q+1:m-1
                        for h=o+1:m
                            for g=h+1:m
                                n=n+1;
                                if m+n<=dimension(2)
                                for i=1:dimension(1)
                        D(i,m+n)=D(i,l)*D(i,k)*D(i,p)*D(i,q)*D(i,o)*D(i,h)*D(i,g);
                                end
                            end
                        end
                    end
                end
            end
        end
    end
 end

end
```

Listing B.9: calculating confluent hypergeometric function

```
%File name: confluenthyper.m
%Input: parameters in the confluent hypergeometric function

function [ new ] = confluenthyper(a,c,x)
%calculate the confluent hypergeometric function with real value
global multiple; %minification
global kk;  %number of minification
n=1;
new=(1+a*x/c)/(multiple^kk);
tempt=[];
old=0;
initialmin=200;
min=initialmin; %minimum number of loops in while loop
while abs(old-new) > 1e-5 || n<min || Fuzhu(a,c,x,n)< Fuzhu(a,c,x,n+1)
        tempt=new;
        new=new+Fuzhu(a,c,x,n);
        old=tempt;
        n=n+1;
end
%prevent the minification to make the value zero
while new == 0
    min=initialmin+min;
    n=1;
    new=(1+a*x/c)/(multiple^kk);
    tempt=[];
    old=0;
    while abs(old-new) > 1e-5 || n<min || Fuzhu(a,c,x,n)< Fuzhu(a,c,x,n+1)
        tempt=new;
        new=new+Fuzhu(a,c,x,n);
        old=tempt;
        n=n+1;
    end
end
```

Listing B.10: calculating terms in the confluent hypergeometric function

```
%File name: confluenthyper.m
%Input: parameters in the confluent hypergeometric function

function [value] = Fuzhu (a,c,x,n)
%calulate terms in the conluent hypergeometric function

global multiple; %minification
global kk;  %number of minification
logvalue=log(a)-log(c);
for i=1:n
    logvalue=logvalue+log(a+i)-log(c+i)-log(i+1);
end
logvalue=logvalue+(n+1)*log(x)-kk*log(multiple);
value=exp(1)^logvalue;
```

References

[1] L. Trocine and L. Malone. An overview of newer,advanced screening methods for the initial phase in an experimental design. In *Proceedings of the 2001 Winter Simulation Conference*, Piscataway, Nj, Aug 2001.

[2] B. Bettonvil and J. P. C. Kleijnen. Searching for important factors in simulation models with many factors: Sequential bifurcation. *European Journal of Operational Research*, 96(1):180–194, 1996.

[3] H. Wan, B. E. Ankenman, and B. L. Nelson. Controlled sequential bifurcation: A new factor-screening method for discrete-event simulation. *Operations Research*, 54(4):743–755, 2006.

[4] R. H. MYers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons Inc., 2002.

[5] T. H. Andres and C. H. Wayne. Using iterated fractional factorial design to screen parameters in sensitivity analysis of a probabilistic risk assessment model. In *Proceedings of the Joint International Conference on Mathematical Models and Supercomputing in Nuclear Applications, 2*, Karlsruhe, Germany, April 1993.

[6] T. H. Andres. Sampling methods and sensitivity analysis for large parameter sets. *Journal of Statistical Computation and Simulation*, 57(1-4):77–100, 1997.

[7] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons Inc., seventh edition, 2008.

[8] G. S. Waston. A study of the group screening method. *Technometrics*, 3(3):371–388, 1961.

[9] C. Mauro. On the performance of two-stage group screening experiments. *Technometrics*, 26(3):255–264, 1984.

[10] C. Mauro and D. E. Smith. Factor screening in simulation: evaluation of two strategies based on random balance sampling. *Management Science*, 30(2):209–221, 1984.

[11] H. Lei, W. G. Pitt, L. K. McGrath, and C. K. Hob. Analysis of supersaturated designs. *Journal of Quality Technology*, 35(1):13–27, 2003.

[12] H. Shen, H. Wan, and S. M. Sanchez. A hybrid method for simulation factor screening. *Naval Research Logistics*, 57(1):45–57, 2010.

[13] R. C. H. Cheng. Searching for important factors: Sequential bifurcation under uncertainty. In *Proceedings of 1997 Winter Simulation Conference*, Piscataway, Nj, 1997.

[14] H. Shen and H. Wan. Controlled sequential factorial design for simulation factor screening. *European Journal of Operational Research*, 198(2):511–519, 2009.

[15] S. H. Kim. Comparison with a standard via fully sequential procedures. *ACM TOMACS*, 15(2):155–174, 2005.

[16] H. Wan, B. E. Ankenman, and B. L. Nelson. Improving the efficiency and efficacy of controlled sequential bifurcation for simulation factor screening. *Informs Journal on Computing*, 22(3):482–492, 2010.

[17] P. McCullagh and J. A. Nelder. *Generalized linear models*. Chapman & Hall/CRC, second edition, 1989.

[18] C. Jennison and B. W. Turnbull. Exact calculations for sequential t, χ^2 and f test. *Biometrika*, 78(1):133–141, 1991.

[19] D.Tang, N. L. Geller, and S. J. Pocock. On the design and analysis of randomized clinical trials with multiple endpoints. *Biometrics*, 49(1):23–30, 1993.

[20] Procedures for comparing samples with multiple endpoints. *Biometrics*, 40(4):1079–1087, 1984.

[21] J. E. Jackson and R. A. Bradley. Sequential χ^2 and t^2-test and their application to an acceptance sampling problem. *Techometrics*, 3(4):519–534, 1961.

[22] M. Beer. Asymptotic properties of the maximum likelihood estimator in dichotomous logistic regression models. Diploma Thesis submitted to the Department of Mathematic, Faculty of Sciences, University of Fribourg Switzerland, Dec 2001.

[23] E. L. Lehmann. *Elements of large-sample theory*. New York: Springer, 1999.