Australian Government
Department of Defence
Science and Technology

Factor Screening Techniques for Combat Simulation Models

Graham V. Weinberg

Joint and Operations Analysis Division
Defence Science and Technology Group

DST-Group–TN–1919

ABSTRACT

A stochastic simulation model can be viewed as a system that takes a series of input variables, and then processes them to produce a realisation of the required simulation scenario output. This model will contain a set of parameters, known as factors, that characterise the simulation process. There may in fact be significant complexity with the underlying simulation architecture, and additionally there may be a very large number of factors. The cost of this is slow performance in simulation run times. In many cases there may be factors in the underlying model that are insignificant statistically in the scenario of interest. This phenomenon is referred to as the parsimony principle, or the Pareto 80-20 rule. Hence there is much interest in the scientific literature on the screening of factors in the underlying meta-model applied to the dynamical system. This report thus investigates the subject of factor screening, for stochastic simulation models, and overviews several solutions to this problem. In particular, sequential bifurcation will be shown to provide a very efficient approach to the problem of factor screening, in comparison to standard one factor at a time classification methods. Several variations of sequential bifurcation will be examined, and their performance in several factor screening examples will be investigated. Directions for possible future research will also be discussed.

Produced by

Joint and Operations Analysis Division
506 Lorimer St,
Fishermans Bend, Victoria 3207, Australia

Telephone: 1300 333 362

Factor Screening Techniques for Combat Simulation Models

Executive Summary

The research project reported here is in support of the Land Capability Analysis (LCA) MSTC's development of an efficient factor screening method to be used with meta-models applied in the analysis of combat simulations generated with computational simulation packages. Combat simulations may involve a large number of factors, that determine various settings of characteristics of the model. In view of the Pareto 80-20 rule, or parsimony principle, only a small number of these factors may have a significant effect on the system response. Hence the challenge is to find an efficient technique or tool to screen these factors, to allow for more efficient use of the simulator.

The report begins by examining the standard testing procedure, where factors are screened one at a time. Then the report examines the process known as sequential bifurcation, where factors can be screened in groups. This procedure is shown to be more efficient than screening factors individually.

In order to account for stochastic variability in the system response, it is necessary to examine sequential bifurcation in the presence of stochatic variability. Hence a series of appropriate techniques are examined. The performance of these will to shown to vary in terms of efficiency and accuracy. Consequently, some rules of thumb will be proposed to facilitate their application in practice.

As a result of these investigations, future research directions will be outlined, that can be used as a basis for further investigations in LCA's combat simulation capability.

This page is intentionally blank

Author

Graham V. Weinberg
Joint and Operations Analysis Division

Graham V. Weinberg is a senior mathematician, who has spent over 16 years in radar signal processing, in the Surveillance and Reconaissance Systems MSTC in ISSD, NSID, and its precursors. His research has supported project AIR 7000, and has spanned the areas of clutter modelling, non-coherent constant false alarm rate radar detector development, coherent multilook radar detection and convergence and approximation problems in radar signal processing. His extensive research in the area of non-coherent radar detection has been documented in a book, published by CRC Press (Radar Detection Theory of Sliding Window Processes, 2017). He serves on the editorial boards of IET Electronics Letters and Elsevier Digital Signal Processing, and is a senior member of the IEEE, as well as a member of the IET.

This report is an account of his research while on an EOI, as an S&T6 Operations Research Specialist, in Land Capability Analysis MSTC in JOAD. Since November 2018 he has been on an EOI in WCSD, as an S&T6 discipline leader in high power RF technologies and effects.

This page is intentionally blank

Contents

Figures

Tables

This page is intentionally blank

1. Introduction

The modelling and simulation of contemporary combat scenarios has become a useful tool for military strategists [1] - [5]. Modern combat scenario simulators, such as COMBATXXI [6], are computationally expensive tools and as such there is much interest in the improvement of their performance. These models have a large number of input parameters, known as factors, and it is of interest to examine whether there are any subsets of these that are in fact redundant, or statistically insignificant. The motivation for screening of factors stems from the well-known parsimony principle, or the famous Pareto 80-20 rule [7], that is the belief that a system's response may in fact only be due to the effects of a smaller subset of input factors. The way in which factor screening is done is to model the entire process as a meta-model, and perform a factor screening analysis to determine whether there are redundancies. This procedure requires evaluation of the simulation model, also keeping the number of simulations of the underlying model to a minimum, while ensuring accurate screening of factors. This approach models the combat dynamical system in a black box construct, and assumptions on the strucure of the meta-model are then applied to study the system response. Such a methodology eliminates the need for analysis of the underlying simulation architecture, and implies that one undertakes a preliminary computer experiment to investigate whether factors are significant, before the results are applied to the simulation model for the generation of desired combat simulations.

The process of formulating and testing whether factors are significant, in a meta-model for some dynamical system, falls into the area of design of experiments (DoE). Three seminal textbooks on this subject matter are [8] - [10]; the latter two were written by those responsible for research developments in the area of factor screening. Other useful summary guides on DoE include [11] and [12].

The application of a meta-model for factor screening analysis is usually done under the assumption that linear combinations of effects is sufficient for modelling the underlying stochastic system. Interactions between factors can also be included, although many of the approaches focus on the main effects model. Interactions can be studied by transformations of the underlying meta-model, that will be discussed subsequently. One common assumption adopted is that interactions are only significant for main effects that are considered significant. This assumption is often used to classify interactions as insignificant. Development of an adequate factor screening method for interactions has not appeared in the literature, and is a potential direction for further investigations.

Some relevant studies of meta-models are [13] - [15]. An assumption of Gaussianity of the errors is usually assumed, that allows the specification of a linear model for the problem. There are justifications, in the assumption of a Gaussian error component, usually on the basis of stationarity of long term averages of processes. It is then possible to propose a factor screening methodology, together with a sensitivity analysis of the factors. With such a methodology it is then possible to reduce the complexity of the simulator by reducing the number of input factors.

There are a large number of approaches for the screening of factors in the context of interest. Early contributions include [16] - [18], who recognised the need for factor screening strategies, for complex simulation models, in a time when computation power was relatively low. These studies applied a random balance/ Plackett-Burman strategy to screen factors. This approach applies a two stage factor screening process. The first stage screens all factors using a random balance approach, while the second stage applies statistical tests on factors determined to be significant in the first stage. Such an approach is not practical in situations where there are a large number of factors, and is essentially an analysis of one factor at a time (OAT). In addition to this, a limitation of the analysis is in the form of meta-model applied to the factor input and output relationship. Specifically, no interactions were examined and so the authors' applied a first order Taylor series approximation to this relationship.

Another early study is [19], known as Morris' approach, that also considered an OAT factor screening, but based upon individually randomised designs and analysis of the resultant sample of observed elementary effects. This screening procedure is based upon the assumption of a deterministic meta-model, and has been the basis for many further investigations. The advantages of Morris' approach is that no explict assumptions are adopted in terms of the number of important factors, nor the requirement of a low order polynomial meta-model approximation. It tends to work well when the number of factors is quite large, but has a larger computational load. Despite these drawbacks, it was demonstrated in [20] that Morris' approach could be applied in the situation where the process is modelling a stochastic combat simulation, thus relaxing the assumption of a deterministic system. In particular, this study applies Morris' OAT factor screening method to SIMBAT, which is a operations analysis tool used by the United Kingdom's Defence Science and Technology Laboratory (Dstl). As discussed in [20], SIMBAT is a closed force-on-force stochastic combat simulator, that allows assessment of equipment performance and combat tactics. Other extensions of Morris' method include [21] and [22], who account for two factor interactions in Morris method, and [23] who introduce a normalisation approach to improve performance. Other contributions include [24] - [28], all of which provide extensions of the original work of [19], including sequential factor screening techniques. The most significant contribution is [27], who demonstrate that false significant factor classification can be minimised, while showing that the modified Morris algorithm is consistent with other sequential factor screening algorithms.

A different line of approach to the problem of factor screening, that has spearheaded much further research, is the idea of applying sequential bifurcation [29, 30]. This technique applies the idea of testing factors for significance as a group. If the original batch of factors is determined to contain at least one significant factor, the group is split into two and each is then tested for the presence of significant factors separately. This process of bifurcating the original group is continued until all factors are classified as significant or redundant. In the presence of sparsely distributed factors, this approach has been shown to improve significantly on Morris's OAT methodology [31, 32]. However, there are a number of limiting factors with the original formulation of this approach. It is assumed that the output process does not contain any significant random error, and that the only significant effects are due to the main effects alone, and so does not account for interactions between factors. In an attempt to formulate sequential bifurcation with a meta-model with significant random error, [33] introduces a version that accounts for model uncertainty. Subsequently, many further studies have extended the method [34] - [37]. An interesting study was presented in [38], who examined a Bayesian approach to the partitioning of bifurcating groups, in an attempt to produce an optimal bifurcating scheme. Further studies of sequential bifurcation and its validation are [39, 40].

Once factors in a simulation model have been screened, it is often of interest to classify the set of important factors via a sensitivity analysis [41, 42]. Two main approaches towards classification of important factors are documented in [41], that is concerned with factor analysis in the context of complex traffic simulation models. The first of these techniques, known as the quasi-optimised trajectory based elementary effects approach, utilises Morris' elementary effects, since it is a measure of local sensitivity. For a given factor a random trajectory is constructed, consisting of elementary effects, and then the mean, absolute mean and standard deviation are used to classify the factors. For example, if the absolute mean is small then the factors are negligible, while if the absolute mean is large and the standard deviation is small then the factors are linear with additive effects but weak interactions. In the case where the absolute mean and the standard deviation are large then the factors exhibit nonlinear effects with possible strong interactions.

The second method for sensitivity analysis of factors is based upon what is known as Kriging [43] - [45]. As reported in [41], this technique is a variance-based methodology, based upon Sobol indicies [46]. This is applied based upon a specific Kriging approximation of the meta-model. Such meta-models assume that the correlation between different samples depend on the distance between input samples [47]. Using a well-known decomposition of variance into components, involving conditional expectations and conditional variance, one can produce a sensitivity index. This has been reported as being a much more robust solution to the classification of the set of important factors [41].

Although sensitivity analysis of factors is an important post-factor screening exercise, it is not the central focus of this report, and is only discussed for completeness. The implementation of the factor screening algorithms was undertaken in MATLAB, and the relevant code can be found in Appendices A to D at the end of the report.

2. Meta-Models

The approach adopted in the analysis to follow assumes a black-box structure for the study of simulation models. This means that the underlying model dynamics is not modelled directly in the factor screening design. Instead a meta-model approach is utilised. Meta-models are higher level designs used to capture the simulation model's outputs, based upon inputs and an assumed relationship between factors. Suppose that the process has a set of N factor effects of interest, which will be denoted $\{\beta_1, \beta_2, \ldots, \beta_N\}$. The system takes as input an N-dimensional vector $\mathbf{x} := (x_1, x_2, \ldots, x_N)$ and produces a response y through some process $\mathcal{S}$, written

$$y = \mathcal{S}(\mathbf{x}|\beta_1, \beta_2, \ldots, \beta_N) := \mathcal{S}(\mathbf{x}), \tag{1}$$

where the dependency on the β_j is suppressed for brevity. It is thus necessary to impose some assumptions on the function $\mathcal{S}$ in order to analyse the importance of factors. This entails assuming that the input and ouput relationship can be modelled by some sort of regression process, that will be referred to as the meta-model.

A main effects meta-model without error, also known as a linearised approximation, assumes that

$$\mathcal{S}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j, \tag{2}$$

where β_j is the jth main effect, β_0 is the overall mean effect and x_j is the jth element of $\mathbf{x}$

$(1 \leq j \leq N)$. Such a model is appropriate in some scenarios that arise in system dynamics and investment analysis. It is more appropriate, when applying a meta-model to some sort of stochastic simulation model, to account for random error in the model (2). A main effects meta-model with error is given by

$$\mathcal{S}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j + \mathcal{E}(\mathbf{x}), \tag{3}$$

where $\mathcal{E}$ is some random process. It is often assumed that this random process has a zero mean and variance of the form

$$\mathrm{Var}(\mathcal{E}(\mathbf{x})) = \sigma^2 R(\mathbf{x}), \tag{4}$$

where $\sigma > 0$ is a constant and $R(\mathbf{x})$ is a nonnegative function of $\mathbf{x}$. Note that the deterministic form of the meta-model has no impact on the variance of the process. Based upon the assumption of stationary processes, and assuming that y is a long term average, it is often assumed that $\mathcal{E}(\mathbf{x})$ is a Gaussian process, that results in tractable solutions. In particular, if the meta-model is assumed to contain an error term that has constant variance then one can construct confidence intervals based upon the assumption of Gaussanity, that can then be used in the classification of the importance of factors.

There are a number of limitations in the assumption of a main effects model that is linear, as in the two models (2) and (3). By eliminating cross-terms and higher order power terms, this process is equivalent to the assumption that a linear Taylor series expansion is a valid approximation for *all* analytic functions. This is clearly invalid in practice. However, one can apply transformations to produce an almost linear realtionship in some functional approximations. Furthermore, without this assumption being imposed, the process of factor screening can become computationally complex. Hence it is necessary to introduce some tradeoffs in order to develop a systematic approach to the problem of factor screening.

Despite this, it is nonetheless possible to account for interactions in the factor screening process, to allow the classification of main effects. This will be demonstrated in subsequent sections. Interactions between factors arise when the factors are inter-related. As an example, a two factor interactions meta-model has the form

$$\mathcal{S}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \beta_{ij} x_i x_j + \mathcal{E}(\mathbf{x}), \tag{5}$$

where the sequence $\{\beta_{ij} : i \in \{1, 2, \ldots, N-1\}, j \in \{i+1, i+2, \ldots, N\}\}$ consists of interaction factors. One can consider other variations; as an example

$$\mathcal{S}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \beta_{ij} x_i x_j + \sum_{j=1}^{N} \beta_{jj} x_j^2 + \mathcal{E}(\mathbf{x}) \tag{6}$$

(Yeesoubi *et al*, 2010). The problem, as noted in the literature, is that increasing the complexity of the interaction model results in significant complexity in the design of an effective and efficient screening process. Some authors have observed that higher order interactions can be ignored because the main effects and lower order interactions are sufficient. Another common assumption, adopted in the analysis of interactions, is that if a factor is not considered important then interactions associated with it are not considered important [48]. Consequently, in the search for important interactions one can restrict attention to interactions associated with main effects classified as important [37]. This is commonly referred to as the hereditary principle in the study of interactions.

The following analysis will be concerned predominantly with factor screening with a main effects meta-model. However, the fold-over design will be introduced and used to test for the importance of main effects in the presence of certain types of interactions.

The next section introduces factor screening for deterministic meta-models, and demonstrates the advantages of sequential bifurcation over OAT factor testing.

3. Factor Screening for Deterministic Meta-Models

Although the context of interest is factor screening for complex simulation models, whose response is subjected to significant random error, it is worthwile overviewing two of the main approaches designed for factor screening for deterministic meta-models. The development of factor screening for the stochastic meta-model case is based upon the approaches developed for the deterministic situation, and so provides a logical starting point. Additionally, this section will demonstrate the merits of utilising sequential bifurcation, in terms of reducing complexity of the factor screening process.

3.1. Morris Elemetary Effects Method

As remarked in Section 1, Max Morris published *Factorial Sampling Plans for Preliminary Computer Experiments* in 1991 [19]. This introduced the idea of applying elementary effects to determine whether factors are important, and is often referred to as a local sensitivity test [41]. One of the advantages of this approach is that it does not require an assumed meta-model. In addition to this it does not require assumptions on the ratio of active to total number of factors, or their effect on the response. It tends to work well when there is a large number of factors, although it assesses the importance of factors OAT, and so can be computationally expensive.

Suppose that there are N factors in the simulation model, scaled to lie in the unit interval. In cases where this is not the case, one can re-parameterise the factor set by dividing each factor by their sum, assuming the latter is positive. If a normalised factor is deemed important then the original factor must also be important. For the purposes of factor screening, each factor is varied across p pre-determined values in [0, 1]. Hence the experimental region is a N-dimensional p-level grid in the product space $[0,1]^N$.

Define $\mathcal{S}(\mathbf{x})$ to be the deterministic output function, for a particular configuration of factors $\mathbf{x} = (x_1, \ldots, x_N)$. Then the elementary effect of the jth factor at $\mathbf{x}$ is defined to be

$$EE_j(\mathbf{x}) := \frac{\mathcal{S}(\mathbf{x} + e_j\Delta) - \mathcal{S}(\mathbf{x})}{\Delta}, \tag{7}$$

where e_j is the unit vector in the direction of the jth axis and Δ is a predefined integer multiple of $\frac{1}{p-1}$, so that $\mathbf{x} + e_j\Delta$ is in the experimental region. Hence the jth factor assumes values in the set $\left\{0, \frac{1}{p-1}, \frac{2}{p-1}, \ldots, 1\right\}$. Roughly speaking, EE_j is the partial derivative of $\mathcal{S}(\mathbf{x})$ with respect to x_j when Δ is small, and so the elementary effect is a local sensitivity measure.

The elementary effect $EE_j(\mathbf{x})$ follows a discrete distribution $\mathcal{F}_j$ that can be realised by randomly sampling the factor combination $\mathbf{x}$. The size of the support of this distribution is $p^{N-1}[p - \Delta(p-1)]$, where p^{N-1} is the number of factor combinations formed by the remaining $N-1$ factors and $p - \Delta(p-1)$ is the number of possible levels that factor j can take to obtain elementary effects. Morris recommends the choice of p be even and $\Delta = \frac{p}{2(p-1)}$. A highly centralised distributional structure for $\mathcal{F}_j$ indicates that the jth factor is important over the experimental region. By contrast, a highly decentralised distribution suggests strong dependence of the jth factor on other factors, that is implying the presence of interactions.

Consequently, Morris' Method determines the importance of the jth factor using the first and second moments of the distribution $\mathcal{F}_j$. In order to achieve this, a series of R trajectories are simulated in the factor product space to produce sample estimates of the first and second moments. Thus if $EE_{j;1}, EE_{j,2}, \ldots EE_{j;R}$ are produced, then the statistics

$$\mu_j = \frac{1}{R}\sum_{i=1}^{R} EE_{j;i} \tag{8}$$

and

$$\sigma_j^2 = \frac{1}{R-1}\sum_{i=1}^{R}(EE_{j;i} - \mu_j)^2 \tag{9}$$

are generated. These estimates are then used to assess the importance of the jth factor. The way in which this is done is to plot a graph of μ_j against σ_j with two boundary lines, corresponding to two standard deviations from the mean: $\mu_j \pm 2\frac{\sigma_j}{\sqrt{R}}$.

If the point (μ_j, σ_j) lies outside the wedged area bounded by the two lines and x- and y-axes, then the jth factor is classified as important. The selection of R is an issue since it needs to be large in order to be accurate, but it is preferential to keep it small in practice.

A simple test example was implemented in Matlab to examine the performance of this algorithm. For this situation the choice of $N = 24$ was adopted, and the meta-model selected has coefficients given in Table 1, that has been adopted from [33]. As reported in the latter, this meta-model is quite challenging to classify. The most significant effects occur at factors 17, 18, and 20 to 24. The mean average effect was selected to be $\beta_0 = 3.2$. Here $p = 4$ was chosen with $\Delta = \frac{2}{3}$. Figure 1 plots the relevant classification figure. The points of interest are those that fall below the boundary line, while those above are considered unimportant and not necessary to classify. The figure indicates that factors β_3, β_{16}, β_{17}, β_{20} and β_{23} can be classified as important, while the other factors are not important. In view of the discussions in [33], there are redundant classifications here. While the method identifies factors 17, 20 and 23 as important, it is likely that factors 3 and 16 are not significant. Additionally, factors 18, 21, 22 and 24 are not identified as important, when in fact it they may be significant in view of the analysis of [33].

Matlab reported that it took 0.018677 seconds to perform the classification. What is interesting about this example is the fact that re-runs of the algorithm result in different classifications

Table 1: Meta-model coefficients, used in the factor screening exercise. The most significant effects are at factor levels 17, 18, 20-24.

j	1	2	3	4	5	6
β_j	5.7	0.6	4.5	19.3	9.1	28.4
j	7	8	9	10	11	12
β_j	51.0	34.6	23.3	39.7	45.4	93
j	13	14	15	16	17	18
β_j	73.7	144.6	130.4	42.5	313.0	166.2
j	19	20	21	22	23	24
β_j	76.6	345.4	195.7	188.8	148.0	206.4

each time. One can increase p in an attempt to compensate for this, but this results in increasing the run time of the algorithm. Further experimentation showed the algorithm became computationally expensive as N increased.

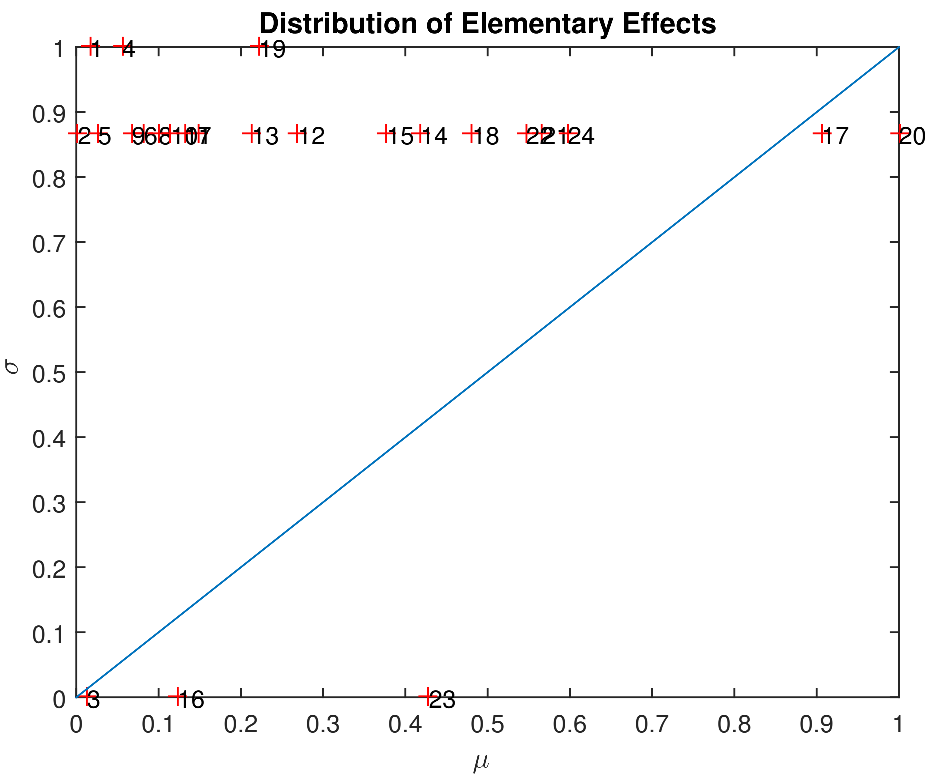

Figure 1: Factor classification diagram produced by Morris method. Important factors appear below the boundary line. Here they are β_3, β_{16}, β_{17}, β_{20} and β_{23}.

3.2. Sequential Bifurcation

Sequential birfurcation was first introduced in [29] in the context of factor screeing for deterministic meta-models. It is essentially an extension of the binary search algorithm, that is used to find an element of an array of a particular rank. Suppose that the simulation output

y is produced by the operation of a function $\mathcal{S}$ acting on the input vector $\mathbf{x}$, so that

$$y = \mathcal{S}(\mathbf{x}), \tag{10}$$

where $\mathbf{x}$ is an N-dimensional vector. Throughout the analysis it will be assumed that N is a power of two; in cases where this is not the case, one can modify the bifurcation process by changing the size of the set of partitioned factors. Alternatively, one can pad out the meta-model with zero factors, that seems to be an acceptable practice provided the zero factors are placed at the end of the meta-model. Hence it is assumed that the meta-model is some N-dimensional polynomial. Here the meta-model is assumed to be deterministic, and such that a rescaling of its input parameters has been performed so that the elements of $\mathbf{x}$ are between zero and unity. A main effects meta-model is assumed for $\mathcal{S}$ so that

$$\mathcal{S}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N} \beta_j x_j, \tag{11}$$

where β_j is the jth main effect, β_0 is the overall mean effect and x_j is the jth element of $\mathbf{x}$ $(1 \leq j \leq N)$. It is assumed that the signs of the main effects are known *a priori*. This assumption is paramount for the sequential bifurcation process to work. In addition, based upon the meta-model (11), it is being assumed that this is a sufficient model for the process (10). That is, the process being modelled does not contain random variation. It is also assumed that $\beta_j \geq 0$, that can be achieved through a reparameterisation of the meta-model. In this scenario a factor is considered significant if $\beta_j > \Phi$, for some threshold $\Phi > 0$. The key observation in group testing is that if the factors are non-negative, and one finds that for a subset of factors $\{\beta_j, j \in \mathcal{J} \subset \{1, 2, \ldots, N\}\}$ that

$$\sum_{j \in \mathcal{J}} \beta_j < \Phi \tag{12}$$

then each $\beta_j < \Phi$, so that one can conclude that this entire set of factors is insignificant and can be eliminated from the meta-model.

To explain this in terms of the meta-model, and to match the original formulation in [29], the case where $\Phi = 0$ is assumed below. Let $y_{(j)}$ be the simulation output of the meta-model, when the first j factors are switched on, while the remaining $N - j$ factors are switched off. Then

$$y_{(j)} = \beta_0 + \beta_1 + \cdots + \beta_j, \tag{13}$$

where $j \geq 2$. Then the sequence induced by $y_{(j)}$ is non-decreasing. Next define $\beta_{j-j'}$ with $j' > j$ to be the sum of effects of factor j through to factor j', so that

$$\beta_{j-j'} = \beta_j + \beta_{j+1} + \cdots + \beta_{j'}. \tag{14}$$

The sequential bifurcation algorithm proceeds as follows. In the initial stage, the two extreme cases of all factors low and all factors high are considered, so that $y_{(0)} := \beta_0$ and $y_{(N)}$ are compared. Hence if there are important factors present then one expects $y_{(0)} < y_{(N)}$; if there are no important factors present then $y_{(0)} = y_{(N)}$. Therefore in the non-redundant case one concludes that $\beta_{1-N} > 0$. Consequently, one splits the set of factors into two sets of length $N/2$, and in the second stage one examines two sets of factors. The first involves comparison of $y_{(N/2)}$ and $y_{(0)}$; if these are equal then one can conclude that there are no significant factors from 1 to $N/2$. As explained in [30], this can result in a dramatic improvement on OAT factor testing, since if there are no important factors in this set, one reduces the number of factors to test by $\frac{1}{2}$. If there is a difference between $y_{(N/2)}$ and $y_{(0)}$ then it is concluded that $\beta_{1-N/2} > 0$ and so there is at least one important factor present in this partition of factors. Continuing on in this manner, this set of factors is then split into two, consisting of $\beta_1, \ldots, \beta_{N/4}$ and $\beta_{N/4+1}, \ldots \beta_{N/2}$. For the first set, one compares $y_{(N/4)}$ with $y_{(N/2)}$. If there is no difference between these, one can conclude that there are no significant factors in the first set, and consequently eliminate them. Otherwise, the process is continued, and is also applies to all other branches of the bifurcating process. This is continued until all factors are classified into important or unimportant categories.

If the collection of important factors is sparse, then such an algorithm can provide rapid classification of important factors, as demonstrated in [30]. Second order interactions can be accounted for by a procedure similar to the foldover principle in [49], that will be discussed subsequently.

Figure 2 provides an example of the way in which sequential bifurcation partitions factors for classification. In this case there are $N = 16$ factors in the meta-model. In stage 0 there is one group only, that is tested for the existence of at least one important factor. In the next stage, this group is partitioned into two groups of size 8, provided stage 0 indicated the existence of important factors. Each group is then tested; a group will be eliminated if it is deemed to have no important factors present. The process then proceeds as shown, until there are at most 16 group of size 1, where each factor has been classified. In this process, once a group of factors is classified as unimportant, it is not further subdivided nor tested. The figure is just to illustrate the way in which the partitioning is performed.

Suppose that in this scenario there are only two important factors, namely β_9 and β_{12}. Figure 3 illustrates the way in which sequential bifurcation will eliminate groups in Figure 2. In the initial stage, the algorithm identifies the existence of important factors, and then divides the factor set into two groups. Stage 1 testing eliminates the first of these sets. This process saves considerably on computational speed relative to OAT factor testing. The next stage then forms two groups of length four, and the test eliminates the second of these. Stage three involves testing of two groups consisting of pairs of factors, with the final result being identification of the two important factors.

A test was conducted where the sequential bifurcation process described above was performed on the same meta-model used to illustrate the classification of factors via Morris' Method. In this case the threshold $\Phi = 200$ was selected. This choice has been motivated on the basis of the discussion of the analysis of the corresponding meta-model in [33]. In order to produce a set of factors that is a power of two, a set of eight zeros were added to the end of the meta-model. Matlab reported that it took 0.001930 seconds to classify the factors, which is an order of magnitude faster than Morris' Method. The algorithm reported that the factors β_{17}, β_{20} and β_{24} were the most significant. Clearly these three factors have the largest coefficient values in the meta-model, and this result is consistent with the results based upon Morris' Method. However, only three factors are deemed important using sequential bifurcation. To imporve this situation, the threshold was reduced to $\Phi = 150$. Matlab then identified the factors 17, 18, 20 to 22 and 24 as important, in 0.001725 seconds.

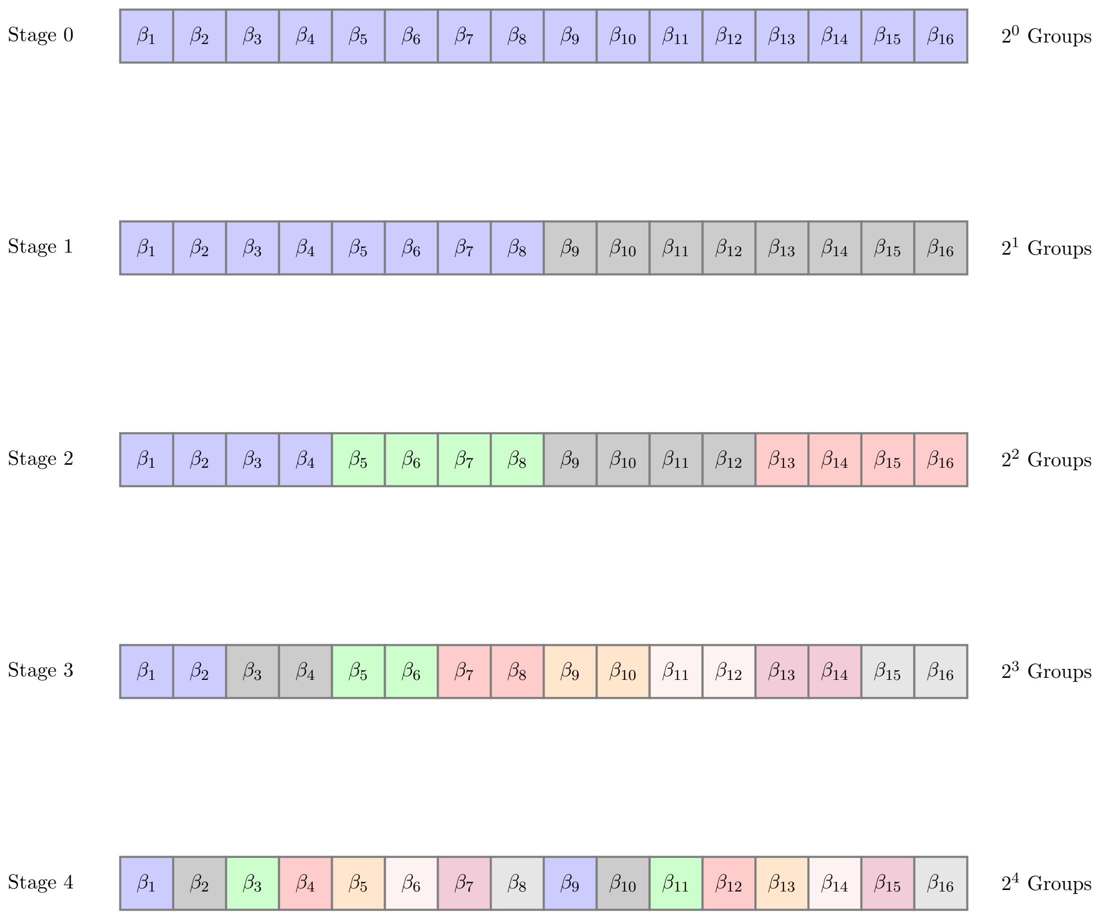
Stage 0
β_1 β_2 β_3 β_4 β_5 β_6 β_7 β_8 β_9 β_{10} β_{11} β_{12} β_{13} β_{14} β_{15} β_{16}
2^0 Groups
Stage 1
2^1 Groups
Stage 2
2^2 Groups
Stage 3
2^3 Groups
Stage 4
2^4 Groups

Figure 2: An illustration of the partitioning of factors into groups during the sequential bifurcation process. In this example there are a total of $N = 16$ factors. The algorithm processes the factors in four stages. In the initial stage, denoted stage 0, the set of factors is tested for the existence of at least one important factor. The algorithm then proceeds by splitting the factors into sets of equal size as shown, which in the maximal case will have the number of groups as indicated in the figure. As the algorithm progresses, groups of factors deemed unimportant will be eliminated, and not tested further.

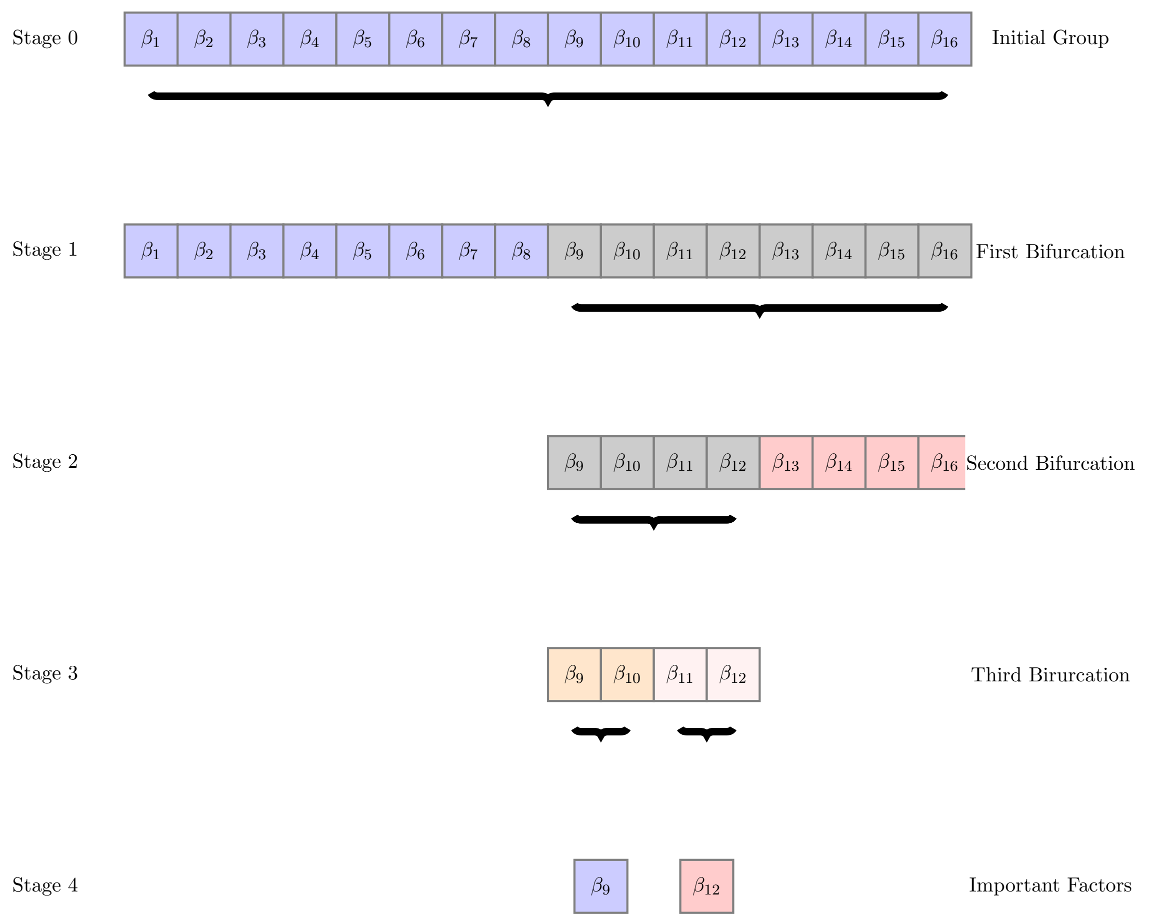
Stage 0
β_1 β_2 β_3 β_4 β_5 β_6 β_7 β_8 β_9 β_{10} β_{11} β_{12} β_{13} β_{14} β_{15} β_{16}
Initial Group
Stage 1
β_1 β_2 β_3 β_4 β_5 β_6 β_7 β_8 β_9 β_{10} β_{11} β_{12} β_{13} β_{14} β_{15} β_{16}
First Bifurcation
Stage 2
β_9 β_{10} β_{11} β_{12} β_{13} β_{14} β_{15} β_{16}
Second Bifurcation
Stage 3
β_9 β_{10} β_{11} β_{12}
Third Birurcation
Stage 4
β_9 β_{12}
Important Factors

Figure 3: An example of the operation of sequential bifurcation, in the deterministic setting, showing the elimination of unimportant factor groups during the bifurcation process. In this example, there are a total of $N = 16$ factors, with two being important. These are β_9 and β_{12}. The sequential bifurcation algorithm begins by identifying the existence of at least one important factor. It then splits the factors into two groups, after which the first is eliminated. The second group is then divided into two, with another group of four factors eliminated. The final group of four factors is then split into two groups, and the two significant factors are then identified.

A key difference with this approach, in contrast to Morris' Method, is that it always produces the same result with each run, meaning that it is more reliable that Morris' Method. Modification of the threshold Φ will result in the classification of other factors as important. The selection of Φ, in the context of meta-models with random error, will be discussed at length in the following section.

A second test was conducted, with a meta-model consisting of $N = 2^{20} = 1,048,576$ factors, with the only non-zero factors being $\beta_{40} = \beta_{200} = 100$, $\beta_{41} = 10$, $\beta_{45} = 15$ and $\beta_{60} = 20$. Although such a large number of factors may not be practicable, this was used to explore the efficiency of the methods and their limits of performance. In this situation Φ was set to unity. It took Matlab 0.029692 seconds to correctly identify all these factors as important. By contrast, Morris' Method took 12.803138 seconds to classify the same meta-model, but when it had its size reduced to $N = 2^{12}$. If N was increased to 2^{20}, the algorithm was terminated after more than an hour, without a final classification of factors.

Thus it is clear that sequential bifurcation can provide an efficient approach for the classification of factors. The next section examines this in the case of stochastic variation in the meta-model.

4. Sequential Bifurcation under Uncertainty

The main conclusion of the previous section is that sequential bifurcation can provide a significant reduction in the time it takes to screen factors, in the deterministic meta-model case. This is certainly the situation when the unimportant factors are clustered, since they can be eliminated in large groups without the need to test OAT. Since the purpose of this research is to establish a suitable factor screening method for COMBATXXI, it is thus necessary to consider meta-models that have a random noise component, such as in (3). Therefore a series of such screening methods will be discussed in this section. Problems experienced with the implementation of these algorithms will also be explored, and remedies adopted will be considered. Matlab code, for all the algorithms discussed in this section, can be found in the appendices.

4.1. Cheng's Method

In 1997 Cheng introduced a version of sequential bifurcation in the case of random error in *Searching for Important Factors: Sequential Bifurcation under Uncertainty* [33]. Basically Cheng examines the same main effects meta-model as in [30], but introduces a zero mean and constant variance Gaussian component, so that the meta-model specified by (3) is assumed. Additional assumptions imposed upon this meta-model is that the sign of the factors are known. Specifically, $\beta_j \geq 0$ for all j is a necessary assumption for the sequential bifurcation process to function.

The approach of Cheng is to identify a statistic, that is a function of factors, such that its mean is a sum of the factors from one level to another. Classification of importance of factors is based upon inclusion in a given region, that can be determined via confidence intervals. Using this test statistic it is possible to eliminate sets of factors as unimportant if they fall outside confidence intervals.

Towards this end, let $I = \{\beta_j : \beta_j > \Phi\}$ and $U = \{\beta_j : \beta_j \leq \Phi\}$. Factors in set I are considered important, while those in U are unimportant. The parameter $\Phi > 0$ is specified by the user, and it is set to zero in the original work in [30]. Appropriate selection of Φ is not really discussed in [33], which means that the user must select this threshold arbitrarily. It will be shown how this threshold can be set automatically, based upon an estimate of factor levels in the assumed main effects meta-model.

Another issue that arises from Cheng's development is the realisation that, due to the stochastic nature of the problem, classification of factors into their sets can be somewhat difficult in cases where a β_j is close to the threshold Φ. In order to address this issue, Cheng introduces an indifference parameter $a > 0$ and in these difficult classification cases, proposes the following solution. If a factor falls in the interval $(0, a + \Phi)$ then it is classified as being in U. Again, selection of the parameter a is based upon the user's discretion, and no systematic or automatic selection of it is discussed in [33].

Cheng reformulates the meta-model (3) in the following way

$$y = \mathcal{S}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N} \beta_j \left(x_j - x_0\right) + \mathcal{E}(\mathbf{x}), \tag{15}$$

where $\mathcal{E}(\mathbf{x}) \overset{d}{=} N(0, \sigma^2(\mathbf{x}))$. The latter is a zero mean Gaussian/Normal distribution with zero mean and variance $\sigma^2(\mathbf{x})$. This is essentially a linearisation around a point $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_N^0)$. In Cheng's approach it is assumed that the meta-model's variance is constant, so that $\sigma^2(\mathbf{x}) = \sigma^2$.

Select a factor index j and make a run with $\mathbf{x}$ set at $\mathbf{x}^{(j)}$ with components

$$\begin{aligned} x_i^{(j)} &= x_i^0 + \delta, i \in \{1, 2 \ldots, j\}; \\ x_i^{(j)} &= x_i^0, i \in \{j+1, j+2, \ldots, N\}, \end{aligned} \tag{16}$$

where $\delta > 0$ is a small quantity one would like to perturb x_i by in a practical application. Denote the output from the simulation run made at this value for $\mathbf{x}$ by $y^{(j)}$, which is referred to as a run made at level j.

For two factor indices $j < k$ define the scaled difference

$$D(j,k) = \frac{[y^{(k)} - y^{(j)}]}{\delta}. \tag{17}$$

It can be shown that

$$\mathbb{E}D(j,k) = \sum_{i=j+1}^{k} \beta_i. \tag{18}$$

Hence $D(j,k)$ can be viewed as an estimator for the sum of factors from $j+1$ to k. With the assumption of Gaussian error with constant variance σ^2 it can be shown that

$$Var(D(j,k)) = \frac{2\sigma^2}{\delta^2}. \tag{19}$$

The main idea is that if the test statistic $D(j,k) < \Phi$ then since the $\beta_i \geq 0$ it follows that each $\beta_i < \Phi$ for $(j+1) \leq i \leq k$. Thus each of these $\beta_i \in U$, and so can be classified as unimportant. Hence one can set up a sequential test, constructing confidence intervals based upon $D(j,k)$, and applying sequential bifurcation as in the original formulation. This then allows one to test for important factors, in a stochastic simulation model, using the basic bifurcation process.

In order to improve the accuracy of the estimator (17), it is possible to reduce its variance through replications. In particular, if one is able to realise n_j replications at level j, namely $\{y_i^{(j)}, i \in \{1,2,\dots,n_j\}\}$, then one can use the average

$$\bar{D}(j,k) = \frac{1}{\delta}\left(\frac{\sum_{i=1}^{n_k} y_i^{(k)}}{n_k} - \frac{\sum_{i=1}^{n_j} y_i^{(j)}}{n_j}\right). \tag{20}$$

Then it can be shown that this is also an unbiased estimator of the sum of main effects from $j+1$ to k, but with variance

$$Var(\bar{D}(j,k)) = \frac{\sigma^2}{\delta^2}\left(\frac{1}{n_j} + \frac{1}{n_k}\right). \tag{21}$$

Based upon (21) one can select n_j and n_k so that this variance is reduced considerably. A simple choice is to choose these to be equal and as large as possible.

Cheng's Method proceeds by constructing one-sided confidence intervals for $\bar{D}(j,k)$, in order to classify the factors in the set $\{\beta_{j+1},\dots,\beta_k\}$. In particular, the test is to classify these factors as unimportant if

$$\bar{D}(j,k) < \Phi - t_\alpha \zeta\sqrt{\frac{1}{n_k} + \frac{1}{n_j}}, \tag{22}$$

where t_α is the $100 \times (1-\alpha)\%$ quantile of a t-distribution with $N-1$ degrees of freedom (corresoponding to the number of replications), and ζ^2 is an estimate of $\frac{\sigma^2}{\delta^2}$. Cheng suggests that the variance be estimated from the sets used during the bifurcation processs, and constructing a pooled estimate of it. However, it was found that this was somewhat problematic in practice. Since Cheng's pooled estimate is a function of the number of bifurcation sets, it was observed that near the end of the sequential bifurcation process, the sets become smaller in size and the variance estimates tended to increase.

In order to circumvent this difficulty, one can utilise the meta-model to obtain an estimate of the population variance quite easily, based upon sampling of the meta-model. Assuming the variance does not vary with the vector $\mathbf{x}$, choose $\mathbf{x}_0 = 0$ without loss of generality. Then evaluating the meta-model at $\mathbf{x} = 0$ results in a realisation of a zero mean Gaussian process with variance σ^2. One can replicate this a number of times and estimate the variance of the resultant sample. This will provide a computationally cheap way in which to estimate the variance.

At the same time one can evaluate the meta-model at $\mathbf{x} = 1$, that is, replacing all values of $\mathbf{x}$ with unity. This results in a realisation of the sum of factors, but offset by a constant (namely β_0) and a zero mean Gaussian process with variance σ^2. Averaging over these realisations provides one with an estimate of the sum of factors. This can then be used to stipulate Φ in Cheng's Method. As an example, one can set Φ to be this mean plus a number of standard deviations. This at least provides an automatic threshold determination method, in contrast to the arbitrary selection of Φ in [33].

Hence one proceeds by testing the significance of groups of factors, much as in the way determinstic sequential bifurcation proceeds. Thus one constructs confidence intervals, classifiying groups of factors until the method results in a series of singleton sets of factors.

The sets of singleton sets are then classified using two-sided confidence intervals, that take the form

$$\bar{D}(j-1,j) \pm t_\alpha \zeta \sqrt{\frac{1}{n_{j-1}} + \frac{1}{n_j}}. \tag{23}$$

If Φ is contained in this interval, then for this particular factor additional replications are performed until a confidence interval can be constructed, such that the following classification can be performed:

1. If Φ is smaller than the lower limit in the confidence interval, then the factor β_j is classified as important;

2. If the upper confidence interval limit is smaller than Φ then the factor β_j is classified as unimportant;

3. if the upper confidence interval limit is smaller than $\Phi + a$, then the factor is classified as unimportant.

At the end of this procedure, each factor classified as important will have an appriopriate confidence associated with it.

Observe that since (17) is an unbiased estimator of the sum of factors in (18), regardless of δ, one can set this to unity without loss of generality. In addition to this, one can choose each $x_i^0 = 0$ in (16), to simplify the evaluation of the meta-model, also without loss of generality.

4.2. Controlled Sequential Bifurcation

Controlled sequential bifurcation was introduced in [35] as an alternative to the factor screening approach of [33], discussed in the previous subsection. The main contribution of [35] was to formulate sequential bifurcation so that the size of the test of classification of factors could be controlled, while the power of the test could also be set to a nominal level. Although [33] provides confidence intervals for classified factors, the method provides no guarantees on the correct classification of factors. Hence [35] provides a significant improvement on [33]. In addition to this, controlled sequential bifurcation is not based upon a constant variance assumption.

Suppose that a factor screening test is such that the null hypothesis is that the factor set under consideration is not important, while the alternative hypothesis is that the factor set contains at least one important factor. Then the size of the test is the probability of classifying a set of factors as important when they are not (Type I error). The power of the test is the probability that a set of factors is classified as important when they actually are significant. The approach in [35] allows these to be set provided that they are complementary (that is, the size is unity less the power). Subsequent to this, [37] then relaxed this condition, allowing the size and power of the tests utilised in sequential bifurcation to be set independently. Thus this section overviews the approaches in [35] and [37].

Controlled sequential bifurcation introduces a two stage classification methodology. It is implemented similarly to Cheng's Method in practice. Suppose that at a given stage of sequential bifurcation, one is testing based upon $\bar{D}(j,k)$; so that one is attempting to classify factors $\beta_{j+1}, \beta_{j+2}, \ldots, \beta_k$. Suppose that $\Psi > \Phi$ is a secondary threshold, to be discussed subsequently. Then define

$$S^2(j,k) = \frac{1}{n_0 - 1} \sum_{i=1}^{n_0} \left(D_i(j,k) - \bar{D}(j,k)\right)^2, \tag{24}$$

which is an estimate of the variance of $D(j,k)$, where n_0 is an initial number of replications. Two thresholds are then required. The first is

$$U(j,k) = \Phi + t_{\sqrt{1-\alpha}} \frac{S(j,k)}{\sqrt{\min\{n_j, n_k\}}}, \tag{25}$$

where n_j and n_k are the number of replications available for the given factor and $t_{\sqrt{1-\alpha}}$ is the $\sqrt{1-\alpha}$ quantile of a t-distribution with $n_0 - 1$ degrees of freedom, and α is the size of the test.

The second threshold is

$$L(j,k) = \Phi - t_{0.5(1+\gamma)} \frac{S(j,k)}{\sqrt{\min\{n_j, n_k\}}}, \tag{26}$$

where n_j and n_k are the number of replications available for the given factor, and $t_{0.5(1+\gamma)}$ is the $0.5(1+\gamma))$ quantile of a t-distribution with $n_0 - 1$ degrees of freedom, and γ is the power of the test, with $\gamma = 1 - \alpha$.

The following integer is also required in the algorithm:

$$N(j,k) = \text{ceil}\left[\frac{h^2 S^2(j,k)}{(\Psi - \Phi)^2}\right], \tag{27}$$

where ceil$[x]$ is the smallest integer exceeding x and h is a constant determined through the equation

$$\mathbb{P}(T \leq t_{\sqrt{1-\alpha}} - h) = \frac{1-\gamma}{2}, \tag{28}$$

where $t_{(\sqrt{1-\alpha})}$ is the appropriate quantile of a t-distribution and T has a t distribution with $n_0 - 1$ degrees of freedom.

Then the controlled sequential bifurcation algorithm tests factors using the following classification scheme, noting that the number of replications at levels j and k are initially $n_j = n_k = n_0$:

1. If $\bar{D}(j,k) \leq U(j,k)$ and $\min(n_i, n_j) \geq N(j,k)$ then the group of factors is classified as unimportant;

2. Else if $\bar{D}(j,k) \leq L(j,k)$ then classify the group as unimportant;

3. Else if $\bar{D}(j,k) \geq U(j,k)$ then classify the group as important;

4. Otherwise generate an additional $|N(j,k) - n_j|$ replications at levels j and k and set $n_j = n_k = \max(N(j,k), n_j)$ so that $D(j,k)$ has an increased number of replications. However, S^2 is not updated, but based upon the initial number of replications. Then

   (a) If $\bar{D}(j,k) < U(j,k)$ then classify the group as unimportant, where $U(j,k)$ is regenerated with a renewed number of replications in its input parameters;

   (b) If $\bar{D}(j,k) \geq U(j,k)$ then the group is classified as important, where the same principle is applied to $U(j,k)$.

The significant feature of this scheme is encapsulated in the following Theorem from [35]:

Theorem 4.1 *Under the assumption of a main effects meta-model (3), with Normally distributed error, sequential bifurcation with the factor classification approach documented above guarantees that the probability of classifying a factor as important, when it is in fact not important, is at most α.*

In addition to this one also has the following guarantee on the power of the test:

Theorem 4.2 *Under the assumptions of Theorem 4.1, the controlled sequential bifurcation procedure guarantees that the probability that a set of factors is classified as important, when their sum exceeds the secondary threshold Ψ, is at least γ.*

Theorems 4.1 and 4.2 imply that the controlled sequential bifurcation algorithm will perform very well in practice. By contrast, [33] does not provide any such guarantees.

As reported in [35], the above sequential bifurcation algorithm is developed under the assumption that the power of the test is complementary to the size. Hence [37] develops an algorithm to relax this assumption. The way in which this is done is to introduce a so-called drift parameter r_0 and test, for given j and k, whether the sum

$$\sum_{i=1}^{r} (D_i(j,k) - r_0) \tag{29}$$

crosses one of two termination boundaries, where r is a number of replications. Depending on which termination boundary is crossed, one can classify the group as either important or unimportant. The algorithm specifies the determination of these boundaries as the solution to a pair of integral equations. Matlab code is provided in an appendix to [37]; however, it was found that these algorithms had implementation issues. Hence an independent implementation in Matlab was performed. Unfortunately it was found that although solutions to these coupled integral equations could be found, there was unacceptable numerical complexity. Hence it was decided that the specialised case of the algorithm in [37] would be investigated, in the situation where the power and size of the statistical test are complementary. Under this restriction it is possible to specify the boundary points of the decision space without solving these integral equations. Hence below this specialised case of the algorithm in [37] is documented.

It is worth noting that such a restriction is not really limiting in practical implementations of controlled sequential bifurcation. For example, if one was to set the size of the test to be

$\alpha = 0.05$, then a test with power $\gamma = 0.95$ is completely acceptable. This is more efficient than increasing the numerical complexity to de-couple the size and power.

Suppose that n_0 is the initial number of replications, as in the standard controlled sequential bifurcation case. Then if the size and power of the underlying test are α and $\gamma = 1 - \alpha$ respectively, define

$$r_0 = \frac{\Phi + \Psi}{2} \tag{30}$$

and

$$a_0 = e^{\left(-\frac{2\log(2\alpha)}{n_0-1} - 1\right)} \frac{(n_0 - 1)}{\Phi - \Psi}. \tag{31}$$

Next specify

$$a(j,k) = a_0 s^2(j,k) \tag{32}$$

and

$$\lambda = \frac{\Phi - \Psi}{4}. \tag{33}$$

Finally define an integer

$$M(j,k) = \text{floor}\left[\frac{a(j,k)}{\lambda}\right], \tag{34}$$

where floor$[x]$ is the largest integer not exceeding x. Then the algorithm proceeds as in previous cases, with the decision rule structure as follows:

1. Set the number of replications at $r = n_j = n_k$, for testing the group of factors $\beta_{j+1}, \beta_{j+2}, \ldots, \beta_k$;
2. If $r > M(j,k)$ then
   (a) If $\sum_{i=1}^{r}(D_i(j,k) - r_0) \leq 0$ then classify the group as unimportant;
   (b) Else the group is classified as important and terminate the classification.
3. If $r \leq M(j,k)$ then
   (a) If $\sum_{i=1}^{r}(D_i(j,k) - r_0) \leq -a(j,k) + \lambda r$ (Termination boundary 1) then the group is classified as unimportant;
   (b) Else if $\sum_{i=1}^{r}(D_i(j,k) - r_0) \geq a(j,k) - \lambda r$ (Termination boundary 2) then the group is classified as important;
   (c) Otherwise add one replication at a time at both levels j and k, increment r to $r+1$ and return to the initial stage of the algorithm.

As in [35], there are guarantees on the success of this algorithm, that are captured in the following result:

Theorem 4.3 *Suppose that the responses $D_i(j,k)$ are independent and identically distributed random variables. Then if $\mathbb{E}D_i(j,k) \leq \Phi$ then the probability of classifying the group as important is at most α. Additionally, if $\mathbb{E}D_i(j,k) \geq \Psi$ then the probability of declaring the group as important is at least γ.*

It is important to note that although Theorem 4.3 provides performance guarantees it is dependent on the number of replications used, to ensure the mean of $D_i(j,k)$ falls within the bounds required. Hence in practical implementations of the method, the success of the classification of factors is a function of the number of replications employed.

4.3. Foldover Design for Interactions

Thus far the factor screening algorithms have been for the case where the meta-model is a main effects model only, as specified by (3). Testing in the presence of interactions has not been examined. One of the innovations in [37] is the introduction of what is termed the foldover design for testing of main effects in the presence of interactions. The method does not allow for the testing of the interations effects. The rationale is that interactions are only important if at least one of its parent factors are important. This is known as the heredity property of factor interactions [48], as discussed previously.

Suppose that the meta-model is of the form

$$Z = \beta_0 + \sum_{i=1}^{N} \beta_i x_i + \sum_{i=1}^{N} \sum_{j=i}^{N} \beta_{ij} x_i x_j + \mathcal{E}(\mathbf{x}), \tag{35}$$

which is a slight modification to the interaction effects in model (5). In addition, Z is used instead of Y in the above to avoid subsequent ambiguity. Then under the assumption of Normally distributed errors with zero mean, one can show that

$$\mathbb{E}D_l(j,k) = \sum_{i=j}^{j} \beta_i + \sum_{i=1}^{j} \sum_{m=j+1}^{k} \beta_{im} + \sum_{i=j+1}^{k} \sum_{m=i}^{k} \beta_{im}. \tag{36}$$

Equation (36) shows that the main group effect is biased by the two factor interactions. This implies that when large negative interactions exist, important main effects may be cancelled out and the screening process inadequate. This problem can be rectified by introducing mirror levels as follows.

Recall that one generates runs of the meta-model via evaluation of (16), where throughout the convention has been to select $x_i^0 = 0$ and $\delta = 1$. Hence $\mathbf{x}^{(j)}$ is a run where the first j elements of the meta-model are switched on and the remaining $N - j$ levels are switched off. Introduce a mirror level by

$$\begin{aligned} x_i^{(-j)} &= -1, i \in \{1, 2 \ldots, j\}; \\ x_i^{(-j)} &= 0, i \in \{j+1, j+2, \ldots, N\}. \end{aligned} \tag{37}$$

Then it can be shown that when one evaluates the meta-model (35) at this mirror level, the response is

$$Z = \beta_0 - \sum_{i=1}^{j} \beta_i + \sum_{i=1}^{j} \sum_{m=i}^{N} \beta_{im} + \mathcal{E}(\mathbf{x}). \tag{38}$$

Hence, one can define a difference of the meta-model

$$Y_l(k) = 0.5(Z_l(k) - Z_l(-k)), \tag{39}$$

where $Z_l(k)$ is the response of the meta-model, at replication l, evaluated with (16) with the appropriate restrictions on parameters applied, and $Z_l(-k)$ is defined similary for the mirror level. One then re-defines

$$D_l(j,k) = Y_l(k) - Y_l(j). \tag{40}$$

Then on the basis of (35) one can show that

$$\mathbb{E}D_l(j,k) = \sum_{i=j+1}^{k} \beta_i. \tag{41}$$

Consequently, one can apply the sequential bifurcation procedure, for a meta-model with interactions as specified in (35), through the transformations induced by (39) and (40), and test for the importance of main effects.

The next section discusses the result of applying these sequential bifurcation algorithms to several test meta-models.

5. Factor Screening Examples

The purpose of this section is to provide some results of the application of the factor screening strategies, discussed in Section 4, when applied to a series of test-case meta-models. These meta-models are variations of the classic form discussed in [33], whose primary meta-model coefficients are provided in Table 1. Since this meta-model is not a power of two it is necessary to introduce extra terms to achieve this. Then variations, via permutations of some of the factors, are investigated to determine whether the algorithm exhibits any sensitivity to the order of factors in the meta-model. In addition to this, a meta-model of significantly larger length is investigated. The following subsection introduces the experimental meta-models used throughout the analysis.

5.1. Test Meta-Models

For the first main effects meta-model, $N = 32$ and $\beta_0 = 3.2$. The meta-model is primarily based upon that in Table 1. The discussions in [30] and [33] suggest that the most significant factors are 17, 20, 21, 22, 23 and 24, in the case of no random error. The issue with the stochastic case is that it is difficult, in the presence of large variance, to classify factors *a priori*. Regeneration of the meta-model will result in further variation and so can result in further classification difficulties. Hence the variance of the meta-model is bounded throughout the following analysis.

The permutation of meta-models will be referred to as a number of cases. Case 1 is the model with coefficients given by Table 1, but with the addition of a series of eight zero terms at the end of the model, to produce a set of $N = 32$ factors. Case 2 is also based upon Table 1, except the final eight factors are set to $\{0.5, 10, 4.6, 25, 20, 3.5, 2, 10\}$. This is to test whether the classification changes if non-zero terms are used to make the meta-model's length a power of two. Case 3 involves a re-ordering of some of the larger main effects, but includes the eight zero terms at the end. Table 2 provides an enumeration of the non-zero factors for clarity. The main idea of this scenario is to investigate whether, if the main effects are placed first in the meta-model, classification is facilitated.

Table 2: Meta-model coefficients for Case 3, which is a re-ordering of the factors in Table 1.

j	1	2	3	4	5	6
β_j	313.0	188.8	148.0	206.4	5.7	0.6
j	7	8	9	10	11	12
β_j	4.5	19.3	9.1	28.4	51.0	34.6
j	13	14	15	16	17	18
β_j	23.3	39.7	45.4	93	73.7	144.6
j	19	20	21	22	23	24
β_j	130.4	42.5	166.2	76.6	345.4	195.7

Table 3: Meta-model coefficients for Case 4, where the meta-model has zero terms at the beginning, together with a slightly large factor of size 100.

j	1	2	3	4	5	6	7	8
β_j	0	0	0	0	0	0	0	100
j	9	10	11	12	13	14	15	16
β_j	5.7	0.6	4.5	19.3	9.1	28.4	51.0	34.6
j	17	18	19	20	21	22	23	24
β_j	23.3	39.7	45.4	93	73.7	144.6	130.4	42.5
j	25	26	27	28	29	30	31	32
β_j	313.0	166.2	76.6	345.4	195.7	188.8	148.0	206.4

The final case, referred to as Case 4, places a series of seven zero factors at the start of the meta-model, followed by a factor of size 100, and then proceeded with the factors in Table 1. Table 3 specifies the meta-model coefficients for clarity. It is of interest to investigate the effect of adding the zero terms initially, together with a slighly large factor, to discover any sensitivities of the classification algorithms.

A second meta-model will also be used, which consists of $N = 1024$ factors, again with $\beta_0 = 3.2$ The first 32 factors are provided in Table 4, while the next 96 factors are random numbers between 1 and 20. The final 896 factors are random numbers in the unit interval. The purpose of this meta-model is to assess the speed with which the factor screening technique operates when the meta-model contains a large number of factors, many of which are in fact redundant.

Each meta-model will have additive Gaussian/Normally distributed error associated with it.

Table 4: The first 32 coefficients of the meta-model of length 1024.

j	1	2	3	4	5	6	7	8
β_j	125.7	0.6	4.5	19.3	9.1	28.4	51.0	34.6
j	9	10	11	12	13	14	15	16
β_j	23.3	39.7	45.4	93	73.7	144.6	130.4	42.5
j	17	18	19	20	21	22	23	24
β_j	313.0	166.2	76.6	345.4	195.7	188.8	148.0	206.4
j	25	26	27	28	29	30	31	32
β_j	0.5	10	4.6	25	20	3.5	2	10

Two primary forms will be used throughout. The first is where the variance is set to a constant. The second case is where it is varied randomly in a prescribed interval. The choice of variance of the Gaussian process will be discussed with each example to follow.

5.2. Cheng's Method

In the first instance, Cheng's Method was run on Case 1 with the variance of the Gaussian component set to unity. The threshold Φ was determined automatically, as discussed in the previous section, while the parameter $a = 5$. For simplicity, the number of replications used throughout the test, for different factors, is set to the same level, so that $n_j = n_k$. The algorithm took 0.188297 seconds, with 50 replications, to identify factors 17, 22, 23, and 24 as significant. The corresponding confidence intervals are $(312.7561, 313.4989)$, $(188.6376, 189.3803)$, $(147.4629, 148.2056)$ and $(205.7679, 206.5106)$ respectively for these factors. These are based upon a test size of $\alpha = 0.05$, so are 95% confidence intervals. As a second test for this case, the variance was varied randomly in the interval $[1, 5]$, to test the algorithm's adaptability to non-uniform variance. The algorithm took 0.135212 seconds to run. The same classification of factors was made, with corresponding confidence intervals $(312.3642, 314.0706)$, $(188.2303, 189.9367)$, $(146.9030, 148.6094)$ and $(205.5725, 207.2789)$ respectively. These confidence intervals are wider due to the change in the variance. In view of the factors in Table 1, Cheng's Method identifies most of the important factors from the deterministic case, but fails to identify factors 20 and 21 as significant. Observe that factor 20 has the largest value, so it is surprising the algorithm fails to identify it. This could be attributed to the addition of random error, or a direct failure of the classification algorithm.

The third experiement considered utilised Case 2 described above for the meta-model coefficients. Again with 50 replications the algorithm took 0.215270 seconds to classify the same set of four factors as important, with confidence intervals $(311.9777, 313.5809)$, $(188.2744, 189.8776)$, $(147.0614, 148.6647)$ and $(205.4632, 207.0664)$. This simulation was for the case where the variance was generated randomly in the interval [1, 5]. Hence it is not necessary to place zeros at the end of the meta-model to increase the length of the meta-model to the appropriate form. Instead one can use smaller valued factors.

Experiment four was based upon Case 3, and with the reordering of factors, the algorithm identified factors 1 to 4 and 21 as important, in 0.218103 seconds. The confidence intervals are $(311.7052, 313.4946)$, $(187.5962, 189.3856)$, $(148.6231, 150.4125)$, $(205.4982, 207.2876)$, $(165.2189, 167.0082)$. It is interesting to observe that factor 23, which has the largest value, is again not identified, while placing significant factors at the beginning of the meta-model results in correct classification of these.

The final experiment, with variations of the same meta-model, involved Case 4, to investigate the effect of padding out the meta-model with zeros initially. With 50 replications the algorithm took 0.197613 seconds to identify the 28th factor as the only important one, with confidence interval $(344.4316, 346.7532)$. Here the most significant factor, from the deterministic case, is identified, while all other significant factors are misclassified. This is an interesting result, since it suggests that placing the redundant terms at the beginning of the meta-model can have an adverse effect on the classification. This may be due to the fact that, since random error is added to the meta-model during its similation, some of these model realisations may in fact be negative values, depending on the factor and variance size. This may then contribute to misclassification. Other simulations found this to be a common problem with the method, and so in order to achieve a factor set of size a power of two, it is advisable to add redundant terms at the end of the meta-model.

Next an example consisting of $N = 1024$ factors was considered. With the Table 4 coefficients first, followed by 96 random numbers between 1 and 20, and 896 random numbers between 0 and 1, the algorithm took 17.748902 seconds to identify factors 1, 18, 20, 21, 23 and 24 as important, with 50 replications. Confidence intervals are $(124.2907, 125.9258)$, $(165.1810, 166.8161)$, $(345.0857, 346.7208)$, $(195.0176, 196.6527)$, $(148.3543, 149.9893)$ and $(205.5573, 207.19240$. Here Cheng's Method identified the most significant factor as important, together with some other significant factors. Thus the method appears to work well when there are a small number of important factors within a large collection of factors. This sparsity effect has been observed in a number of publications [30, 33].

5.3. Controlled Sequential Bifurcation I

The same set of experiments are now tested using the first version of controlled sequential bifurcation, based upon [35]. The size of the test has been set to $\alpha = 0.05$, so that the power of the test is $\gamma = 0.95$. The selection of the threshold Φ has been on the basis of an average effects estimate from the meta-model, as in the implementation of Cheng's approach. The choice of $\Psi = \Phi + 20$ has been applied, based upon similar choices in [35]. Throughout the following the error's variance is generated as a random number in the interval [1, 5].

For the simulation of Case 1, it was found that 50 replications identified factor 24 as the only important one, taking 0.423154 seconds to run. Increasing the number of replications to 100 took 0.761860 seconds to identify factors 22 to 24 as significant. 200 replications, taking 1.289307 seconds, identified factors 17, 21 to 24 as important. These are most of the significant factors, for the deterministic case, but factor 20 is not identified, which has the largest effect. It took 1000 replications to identify factors 15, 18 to 24 as important, which now includes factor 20, but loses factor 17 in the classification. This took Matlab 5.365446 seconds to perform.

Next test Case 2 was examined, and it took the algorithm 500 replications, in 2.99 seconds, to classify factors 17, 20-28 as important. Here the most significant factors, from the deterministic case, are identified, together with some possible misclassifications.

A simulation of Case 3, with 500 replications, took the same time to identify factors 1 to 4, 19 and 22-24 as important. Examining the meta-model coefficients in Table 2, one observes that all of the significant factors from the deterministic case are identified, together with a possible single misclasssification.

The final test with permutations of this meta-model is the situation of Case 4, where zeros are placed at the beginning of the meta-model. The same problem, as with Cheng's Method, was observed in this scenario. With 500 replications no significant factors could be identified. Increasing this to 1000 replications resulted in the same issue. Hence this algorithm has a sensitivity to the order in which redundant factors are placed in the meta-model.

To complete the analysis of this algorithm, the meta-model with $N = 1024$ factors was investigated, with the same test scenario as for Cheng's Method. Using 100 replications it took 127.821918 seconds to identify factors 1 and 18 to 24 as important. Here most of the important factors are identified, with the exception of factor 17, and some misclassifications have occured. It is interesting to observe that Cheng's Method did a slighly better classification in this case, in a smaller run time.

5.4. Controlled Sequential Bifurcation II

Next the algorithm adopted from [37] is investigated with these experimental test cases. Again, the same noise settings and threshold determinations are adopted from the previous subsection.

For Case 1, with an initial minimum of 500 replications, Matlab took 1.713891 seconds to identify factors 15,16, 18 to 24 as important. This classification did not identify factor 17, although it identified the most significant factor from the deterministic case. Increasing the number of replications to 1000 took 3.712761 seconds to identify factors 10 to 12 and 14 to 24 as important. Here all the significant factors are identified, but with some possible misclassifications.

Test Case 2 was performed with 500 replications, and took 2.152188 seconds to classify the same set of factors as important. It is interesting to note that the classification was faster with zeros used at the end of the meta-model.

For the scenario of Case 3, it took 500 replications in 1.739505 seconds to identify factors 1 to 4, and 19, 20, 22 to 24 as important. The most significant factors are identified as important with the reordering, together with some possible misclassifications.

Case 4 exhibited the same issues as with the other two classification tests, and so again placing zeros at the beginning of the meta-model resulted in issues with classification.

For the final experiment with this version of sequential bifurcation, the meta-model with $N = 1024$ factors was tested. 100 replications took 99.618817 seconds to identify factors 1, 18 to 24 as important. This was a better result than that with the first version of sequential bifurcation, but not as fast as Cheng's Method.

5.5. Summary of Results

For convenience the results in Subsections 5.2 - 5.4 are summarised in the following two tables. In the first instance, Table 5 includes a comparision of factor screening results for Cases 1 to 3; results for Case 4 are excluded since the main conclusion from this is that placing zeros at the start of the meta-model, to achieve a power of two, is detrimental to the screeing method. The results in Table 5 are for the meta-models of length $N = 32$.

Table 6 provides a comparison of the three factor screeing methods for the meta-model with $N = 1024$.

Table 5: Illustration of the main results of factor classifiication for the case of $N = 32$. For each of the three non-terminal cases, the best results with each of the factor screening methods is included. In each case, the identified important factor set is included, as well as the number of replications and the time taken to run the algorithm.

	Cheng	CSB I	CSB II
Case 1	$\{17, 22, 23, 24\}$ 50 reps 0.135212 secs	$\{15, 18, 19, \dots, 24\}$ 1000 reps 5.365446 secs	$\{15, 16, 18, 19, \dots, 24\}$ 500 reps 1.713891 secs
Case 2	$\{17, 22, 23, 24\}$ 50 reps 0.218103 secs	$\{17, 20, 21, \dots, 28\}$ 500 reps 2.99 secs	$\{10, 11, 12, 14, 15, \dots, 24\}$ 500 reps 2.152188 secs
Case 3	$\{1, 2, 3, 4, 21\}$ 50 reps 0.218103 secs	$\{1, 2, 3, 4, 19, 22, 23, 24\}$ 500 reps 2.99 secs	$\{1, 2, 3, 4, 19, 20, 22, 23, 24\}$ 500 reps 1.739505 secs

Table 6: Summary of factor screening results for the meta-model with $N = 1024$ factors. The results show, for the particular meta-model considered, the identified important factors obtained with each of the methods.

Cheng	CSB I	CSB II
$\{1, 18, 20, 21, 23, 24\}$ 50 reps 17.748902 secs	$\{1, 18, 19, \dots, 24\}$ 100 reps 127.821918 secs	$\{1, 18, 19, \dots, 24\}$ 100 reps 99.618817 secs

5.6. Controlled Sequential Bifurcation with Interactions

This last subsection provides some examples of main effects classification in the presence of interactions. Only the main effects model, based upon Case 1, is considered. For the first series of examples, the meta-model used takes the form

$$\mathcal{S}(\mathbf{x}) = 3.7 + \sum_{j=1}^{32} \beta_j x_j + \tau_1 \sum_{i=1}^{32} x_{i-1} x_i + \tau_2 \sum_{i=2}^{32} x_i^2 + \mathcal{E}(\mathbf{x}), \tag{42}$$

where the β_j are as specified in Table 1, with a series of zero terms added to make it a power of two. This simplified model has been employed to facilitate the numerical experimentation. This is a main effects meta-model with second order interactions spanning across almost the complete set of main effects. For simplicity interactions occur at common levels as given by the coefficients τ_1 and τ_2. Similar simulation parameters are chosen as in the previous cases. In particular, the variance of the Gaussian component is generated uniformly in the interval [1, 5] throughout.

For the first scenario, $\tau_1 = 3$ and $\tau_2 = 5$. Using 500 replications, Matlab took 15.371709 seconds to classify factors 15, 16, and 18 to 24 as important. Most of the important factors, from the deterministic case, have been identified as important. When the number of replications is increased to 1000, it took 34.596816 seconds to classify factors 11, 12, and 14 to 24 as important. Increasing the number of replications facilitated the identification of the most important factor from the deterministic case, namely factor 20.

Next the interaction parameters are increased to $\tau_1 = 100$ and $\tau_2 = 200$ repectively. This is in an attempt to induce strong interactions, and to assess the classification of main effects in the presence of such interactions.

500 repetitions took 14.452771 seconds in Matlab to classify the main effects 17, and 22 to 24 as important. Increasing the number of replications to 1000 resulted in the classification of factors 17, and 20 to 24 as important. This took 29.097572 seconds in Matlab to achieve; here all the significant main effects, from the deterministic case, have been identified as important. This implies that the foldover design is working well to eliminate the interactions and permitting the classification of important main effects.

Consider next the meta-model specified by

$$\mathcal{S}(\mathbf{x}) = 3.7 + \sum_{j=1}^{32} \beta_j x_j + \sum_{i=1}^{32} (10+i) x_{i-1} x_i + \sum_{i=2}^{32} (5+i) x_i^2 + \mathcal{E}(\mathbf{x}), \tag{43}$$

where the main effects are as for the previous example. With 500 replications Matlab took 15.148977 seconds to classify factors 15, and 18 to 24 as important. This test has identified most of the important factors from the deterministic case. When the number of replications is increased to 1000 it took 34.606846 seconds to classify factors 11, 12, and 14 to 24 as important. Here all the important factors, from the deterministic case, are identified, together with some other factors. These are either becoming important, due to the interaction effect, or are a possible misclassification, due to insufficient number of replications.

The third interaction analysis is based upon the meta-model specified by

$$\mathcal{S}(\mathbf{x}) = 3.7 + \sum_{j=1}^{32} \beta_j x_j + \sum_{i=1}^{32} (100+i) x_{i-1} x_i + \sum_{i=2}^{32} (50+i) x_i^2 + \mathcal{E}(\mathbf{x}), \tag{44}$$

which has stronger interactions than the previous case, but which still vary with the interaction index. As before, the main effects are as those specified in Case 1 previously. It took 14.915053 seconds to classify factors 17, and 21 to 24 as important, with 500 replications. When the number of replications is increased to 1000, it took 30.210147 seconds to identify factors 15, 16, and 18 to 24 as important. Most of the important main effects, from the deterministic case, have been identified as important.

A fourth study of the effects of interactions applies the meta-model

$$\mathcal{S}(\mathbf{x}) = 3.7 + \sum_{j=1}^{32} \beta_j x_j + \sum_{i=21}^{24} (100+i) x_{i-1} x_i + \sum_{i=21}^{24} (50+i) x_i^2 + \mathcal{E}(\mathbf{x}), \tag{45}$$

which limits the interactions of meta-model (44) to a smaller set of indices. Here the same main effects model, from Case 1, is applied. Matlab took 15.089950 seconds to classify factors 15, 16, and 18 to 24 as important, using 500 replications. When the number of replications is increased to 1000, it took 33.294631 seconds to classify factors 11, 12 and 14 to 24 as important.

For the reader's benefit, Table 7 provides a summary of the results presented in this subsection. In all cases 500 replications have been utilised, and so this specification is thus omitted from the table.

Table 7: Comparison of the factor screening of main effects, in the case of interactions. The table summarises the model examined, together with the main factors classified as important and the time taken to perform the classification.

Model	Results
(42), $\tau_1 = 3$, $\tau_2 = 5$	$\{11, 12, 14, 15, \ldots, 24\}$ 34.596816 secs
(42), $\tau_1 = 100$, $\tau_2 = 200$	$\{17, 20, 21 \ldots, 24\}$ 29.097572 secs
(43)	$\{11, 12, 14, 15, \ldots, 24\}$ 34.606846 secs
(44)	$\{15, 16, 18, 19, \ldots, 24\}$ 30.210147 secs
(45)	$\{11, 12, 14, 15, \ldots, 24\}$ 33.294631 secs

6. Summary and Future Directions for Research

The purpose of this study was to identify a suitable factor screening method, to be used with meta-models applied to combat simulation models. If the number of factors is not too large, and their signs could not be assumed to be homogeneous, then one could apply Morris' Method to provide a series of elementary effects, and allow the identification of important factors. Here the definition of not too large is equivalent to assuming the number of factors is not in the thousands, but more in the order of hundreds. When the preliminary classification of factors is undertaken with Morris' Method, one could then re-construct a meta-model, based upon factors of similar signs, and then run a sequential bifurcation algorithm to further classify like-signed factors.

Under the assumption of a main effects model with non-negative factors, it is clear that Cheng's Method can classify important factors adequately, when there are a large number of redundant factors present in the meta-model. If the number of factors is moderate, then either of the controlled sequential bifurcation algorithms can be applied. In the presence of interactions it is clear the second version of sequential bifurcation, adapted to manage second order interactions, can classify factors very well. One of the *rules of thumb* of the sequential bifurcation process, in the presence of uncertainty, is that the classification process is facilitated if the important main effects are clustered near the beginning of the meta-model. Hence one can arrange this, on the basis of a user's belief of which factors may be important.

This research project has indicated that there are a number of avenues available for future research. One of the primary issues observed in other simulation runs not reported here is that as the error's variance increases, the classification of factors becomes much more difficult. When the variance is not too large relative to the meta-model's main effects, successful classification is possible. Hence it would be useful if one could reduce the variance of the noise in practice. This is not an unrealistic expectation. Note that one can view a main effects meta-model, such as (3), for given $\mathbf{x}$ and factors β_j, as what engineers view as a constant signal embedded in additive Gaussian noise (see for example [50]). Reducing the variance of the error would essentially make the sequential bifurcation process one of classifying factors in negligible noise. The latter can be done very efficiently, as discussed previously. Hence one would like to make the main effects component, given by (2), much larger relative to the error component, in order to improve the classification process.

A matched filter is used by engineers for such purposes. This is a linear filter that maximises the average power of the signal, relative to the average power of the noise, in its output [52]. Hence it is believed that a matched filter could be applied during the sampling of the meta-model, in such a way that it essentially reduces the noise's variance. In order to do this it will be necessary to understand the effect the linear transformation has on the set of factors. This is certainly a possible solution for factor screening in the presence of errors with large variance. A useful discussion of the effects of a matched filter on signals in Gaussian interference is [51].

A second development, that merits further research, is to investigate the sensitivity of the classification process to the presence of zero factors in the meta-model. Recall that these zero terms were used to produce a set of factors that is a power of two. The observed misclassification resulting from zero factors placed at the beginning of the meta-model may be due to the fact that runs of it may yield negative values, for the test statistics used in the testing of factor groups. An alternative solution is to redesign the sequential bifurcation process to operate in situations where the number of factors is not a power of two.

Finally, it may be possible to develop a capability to screen interactions in groups, by observing that if in (39), the minus sign is replaced with a plus, then it is possible to eliminate the main effects, although the term β_0 is retained. However, when one re-calculates (40), under this conditions, then β_0 will be eliminated. What remains is a set of interaction effects. Hence one could initiate the screening of interactions based upon such a transformation. It is recommended that the interested reader begin with a simple interaction model such as (42), produce the alternative to (40) and then attempt to classify the interactions.

Acknowledgements

The author is grateful for the opportunity provided by Dr Andrew Gill and Dr Fred Bowden to spend six months in LCA, and providing the author with his first experience working at the S&T6 level. Comments and technical review of the report were also appreciated.

References

[1] Kilmer, R. A. Application of Artificial Neural Networks to Combat Simulations. Mathematical Compuer Modelling, 23, 91-99, 1996.

[2] Sanchez, S. M., Lucas, T. W., Sanchez, P. J., Nannini, C. J., Wan, H. Designs for Large-Scale Simulation Experiments, with Aplications to Defense and Homeland Security. Naval Postgraduate School Preprint, Monterey, 2012.

[3] Xing, D., Wan, H., Zhu, M. Y. Simulation Screening Experiments using Lasso-Optimal Supersaturated Design and Analysis: A Maritime Operations Application. Proceedings of the 2013 Winter Simulation Conference, 497-508, 2013.

[4] Boutselis, P., Ringrose, T. J. GAMLSS an Neural Networks in Combat Simulation meta-modelling: A Case Study. Expert Systems with Applications, 40, 6087-6093, 2013.

[5] Cioppa ,T., Brown, L. Agent Based Models as an Exploratory Analysis Approach to Combat Simulations. US Army TRADOC Analysis Center, Monterey, California 2002.

[6] CombatXXI Description, http://www.trac.army.mil/MS_TRAC.html. last accessed June 2018.

[7] Juran, J. M. Management of Quality Control. New York, 1967.

[8] Montgomery, D. C. Design and Analysis of Experiments. John Wiley, New York, 2001.

[9] Kleijnen, J. P. C. Design and Analysis of Simulation Experiments. Springer, 2008.

[10] Morris, M. D. Design of Experiments: An Introduction Based on Linear Models. Chapman and Hall/CRC Press, 2017.

[11] Yaesoubi, R. A Comparison of Factor Screening Methods for Simulation Models. Master of Science Thesis, North Carolina State University, 2006.

[12] Woods, D. C., Lewis, S. M. Design of Experiments for Screening. ArXiv Preprint, 1510.05248v1, 2015.

[13] Brockwell, S. E., Gordon, I. R. A Comparison of Statistical Methods for Meta-Analysis. Statistics in Medicine, 20, 825-840, 2001.

[14] De la Fuente, R., Smith, R. Metamodelling a System Dynamics Model: A Contemporary Comparison of Methods. Proceedings of the 2017 Winter Simulation Conference, 1926-1937, 2017.

[15] Nagashima, K., Noma, H., Furukawa, T. A. Prediction Intervals for Random Effects Meta-Analysis: A Confidence Distribution Approach. ArXiv Preprint, 1804.01054v3, 2018.

[16] Smith, D. E., Mauro, C. A. Factor Screening in Computer Simulations. Simulation, 38, 49-54, 1982.

[17] Mauro, C. A., Smith, D. E. Factor Screening in Simulation: Evaluation of Two Strategies Based on Random Balance Sampling. Management Science, 30, 209-221, 1984.

[18] Sacks, J., Welch, W. J., Mitchell, T. J., Wynn, H. P. Design and Analysis of Computer Experiements. Statistical Science, 4, 409-435, 1989.

[19] Morris, M. D. Factorial Sampling Plans for Preliminary Computational Experiments. Technometrics, 33, 161-174, 1991.

[20] Alam, F. M., McNaught, K. R., Ringrose, T. J. Using Morris' Randomized OAT Design as a Factor Screening Method for Developing Simulation Metamodels. Proceedings of the 2004 Winter Simulation Conference, 2004.

[21] Campolongo, F., Braddock, R. The Use of Graph Theory in the Sensitivity Analysis of the Model Output. Reliability Engineering and System Safety, 64, 1-12, 1999.

[22] Cropp, R. A., Braddock, R. D. The New Morris Method: An Efficient Second Order Screening Method. Reliability Engineering and System Safety, 78, 77-83, 2002.

[23] Campolongo, F., Cariboni, J., Saltelli, A. An Effective Screening Design for Sensitivity Analysis of Large Models. Environmental Modelling and Software, 22, 1509-1518, 2007.

[24] Boukouvalas, A., Gosling, J. P., Maruri-Aguilar, H. An Efficient Screening Method for Computer Experiments. Technometrics, 56, 422-431, 2014.

[25] Fedou, J. M., Rendas, M. J. Extending Morris Method: Identification of the Interaction graph Using Cycle-Equitable Designs. Journal of Statistical Computation and Simulation, 85, 1398-1419, 2015.

[26] Khare, Y. P., Munoz-Carpena, R., Rooney, R. W., Martinez, C. J. A Multi-Criteria Trajectory-Based Parameter Sampling Strategy for the Screening Method of Elementary Effects. Environmental Modelling and Software, 64, 230-239, 2015.

[27] Shi, W., Chen, X., Shang, J. An Efficient Morris Method Based Framework for Simulation Factor Screening. Technical Report, Department of Industrial and Systems Engineering, Virginia Tech, USA.

[28] Shi, W., Chen, X. Controlled Morris Method: A New Distribution-Free Sequential Testing Procedure for Factor Screening. Proceedings of the 2017 Winter Simulation Conference, 1820-1831, 2017.

[29] Bettonvil, B. Detection of Important Factors by Sequential Bifurcation. PhD Thesis, Tiburg University, 1990.

[30] Kleijnen, J. P. C., Bettonvil, B. W. M. Searching for Important Factors in Simulation Models with Many Factors: Sequential Bifurcation. European Journal of Operations Research, 96, 180-194.

[31] Kleijnen J. P. Factor Screening in Simulation Experiments: Review of Sequential Bifurcation. In: Alexopoulos C., Goldsman D., Wilson J. (eds) Advancing the Frontiers of Simulation. International Series in Operations Research & Management Science, vol 133. Springer, Boston, MA, 2009.

[32] Kleijnen, J. P. Sensitivity Analysis of Simulation Models. CentER Discussion Paper, Tilburg University, 2009.

[33] Cheng, R. C. H. Searching for Important Factors: Sequential Bifurcation Under Uncertainty. Proceedings of the 1997 Winter Simulation Conference, 275-280, 1997.

[34] Shen, H., Wan, H. Controlled Sequential Factorial Design for Simulation Factor Screening. Proceedings of the 2005 Winter Simulation Conference, 467-474, 2005.

[35] Wan, H., Ankenman, B., Nelson, B. L. Controlled Sequential Bifurcation: A New Factor Screening Method for Discrete Event Simulation. Operations Research, 54, 743-755, 2006.

[36] Yaesoubi, R., Roberts, S. D., Klein, R. W. A Modification of Cheng's Method: An Alternative Factor Screening Method for Stochastic Simulation Models. Proceedings of the 2010 Winter Simulation Conference, 1034-1047, 2010.

[37] Wan, H., Ankenman, B. E., Nelson, B. L. Improving the Efficiency and Effficacy of Controlled Sequential Bifurcation for Simulation Factor Screening. INFORMS Journal on Computing, 22, 482-492, 2010.

[38] Frazier, P. I., Jedynak, B., Chen, L. Sequential Screening: A Bayesian Dynamic Programming Analysis of Optimal Group-Splitting. Proceedings of the 2012 Winter Simulation Conference, 2012.

[39] Shi, W., Kleijnen, J. P. C. Validating the Assumptions of Sequential Bifurcation in Factor Screening. Discussion Paper 2015-034, Tilburg University, Center for Economic Research, 2015.

[40] Shi, W., Kleijnen, J. P. C. Testing the Assumption of Sequential Bifurcation for Factor Screening. Simulation Modelling Practice and Theory, 81, 85-99, 2018.

[41] Ge, Q., Ciuffo, B., Menendez, M. An Exploratory Study of Two Efficient Approaches for the Sensitivity Analysis of Computationally Expensive Traffic Simulation Models. IEEE Transactions on Intelligent Transportation Systems, 15, 1288-1297, 2014.

[42] Lam, H. Robust Sensitivity Analysis for Stochastic Systems. ArXiv Preprint, 1303.0326v2, 2015.

[43] Krige, D. G. A Statistical Approach to some Mine Valuations and Allied Problems at the Witwatersrand. Master's thesis of the University of Witwatersrand, 1951

[44] Pousi, J., Poropudas, J., Virtanen, K. Game Theoretic Simulation Metamodelling using Stochastic Kriging. Proceedings of the 2010 Winter Simulation Conference, 1456-1467, 2010.

[45] Mehdad, E., Kleijnen, J. P. C. Stochastic Intrinsic Kriging for Simulation Metamodeling. Applied Stochastic Models in Business and Industry, 34, 322-337, 2018.

[46] Sobol, I. M. Sensitivity Estimates for Nonlinear Models. MMCE Vol 1, 407-414, 1993.

[47] Stein, M. L. Statistical Interpolation of Spatial Data: Some Theory for Kriging. Springer, New York, 1999.

[48] Wu, J. C. F., Hamada, M. Experiments: Planning, Analysis and Parameter Design Optimization. John Wiley and Sons, New York, 2000.

[49] Box, G. E. P. and Wilson, K. B. On the Experimental Attainment of Optimum Conditions. Journal of the Royal Statistical Society, Series B, 13, 1-38 , 1951.

[50] Levanon, N. Radar Principles, J. Wiley and Sons, (Interscience Div.) New York, 1988.

[51] Levanon, N. and Mozeson, E. Radar Signals. J. Wiley and Sons (Interscience Div.), New York, 2004.

[52] Richards, M. A. Fundamentals of Radar Signal Processing. McGraw-Hill, 2014.

Appendix A. Cheng's Method

A.1. Main Algorithm

```
%This code provides an implementation of Cheng's SB under uncertainty. Note
%that Ceng's method is designed for the case of constant variance in the
%meta-model, although it appears to work reasonably well with slight
%variation in the distribution's variance. Cheng's method has been modified
%for the estimation of variance. Cheng uses a weighted pooled estimate
%adaptively, which tended to affect performance when the number of
%significant factor in the meta model reduced. Hence the approach is to
%estimate the variance initially from the meta model, which is not
%computationally expensive. Cheng's parameter B, which is used as a
%threshold, can be estimated from the meta-model too; otherwise it is
%chosen on the basis of a priori knowledge of the level of importance of
%factors. The indiffference parameter a used in the classification of
%boundary important factors is chosen in an ad hoc fashion. Cheng's paper
%provides very little guidance on the selection of B and a in practice.
%
%
%The number of factors in the meta-model N must be set to match the meta
%model manually.
%
%
%With reference to the formulation in Cheng's paper, the following changes
%are also adopted. The low fixed level $x_i^0$ can be set to zero without
%loss of generality. The parameter $\delta$ can be set to unity. Hence
%D(j,k)$ is the difference of two runs of the meta model, with the first
%component being when the first $k$ levels are switched on, and the second
%being the case where the first $j$ levels are switched on (k>j). D(j,k) is
%still an unbiased estimator of the sum of factors from j+1 to k regardless
%of $\delta$. Note that in the code below D(j-1, k) is used since it is an
%unbiased estimator of the sum of factors from j to k.
%


clear all;

close all;



N = 32;
%Number of factors; this must be matched to the meta-model, and must be a power of 2.


test_size = 0.05;
%Size of test, to determine the quantile for the threshold and confidence intervals.


a = 5;
%This parameter is a bit ad hoc in Cheng's paper.

noreps = 50;
%Number of replications; the first is for the determination of D(i, j) via regeneration of the meta-model.
%This is the average on the bottom of page 277 in Cheng's paper. In principle one can have two separate number of replications.
%However, for this work it was decided to keep $r^{(k)}$ and $r^{(j)}$ equal.
noreps2 = 50;
%This is for the determination of the mean and variance of the meta-model.

m = log2(N);
%power of two. If the meta-model is not a power of two one can pad it out with zeros, provided they occur at the end of the model.
%Numerical experimentaton showed that if put at the start, it can cause a serious misclassification. The report explains why this happens.


factor_index_vector = 1:N;
%This vector stores the factor index of important factors as SB proceeds.
%Hence the jth entry of this is the index of the jth remaining factor as the algorithm proceeds.
%The length of this vector will change dynamically as the SB algorithm progresses.

for kdx=1:noreps2
    res1(kdx) = test_meta_model(zeros(1,N));
    res2(kdx) = test_meta_model(ones(1,N));
end

var_est = var(res1);

B = mean(res2)/noreps2;% -0.1*sqrt(var_est);
%This is an attempt to make this selection automatic. Cheng just chooses it based upon a guess.
%But it seems more logical to base it upon an estimate of the mean of the factors from the meta-model.
% One can offset this by the variance if desired.


%The above estimates the common variance using the meta model. This takes very little
%time to run, and is better than Cheng's procedure of estimating it on the
%basis of factor index sets. Implementation of the latter resulted in very
%large variance estimates, which became worse as the size of the factor
%sets reduced. Basically this is small sample bias. One would need to add
%many more replications to do it via Cheng's approach. So it is more
%efficient to estimate it from the meta model.
```

```
%The algorithm begins by assesssing whether there is at least one important
%factor.

test_statN = simulateD(0, N, N, noreps2, noreps2);
test_stat_0 = sum(res1)/noreps2;   %This is matched to test_stat2 indirectly.


if test_statN - test_stat_0  <= B - tinv(1-test_size, N-1)*sqrt(var_est)*sqrt(2/noreps2);  %No important factors at all

   disp('No important factors');

else %There is at least one important factor, so begin the analysis.
%This works by partitioning factor_index_vector into subgroups and then retaining sets of indices corresponding to groups of important factors.


for jdx=1:m              %At the jth stage the algorithm is classifying groups of size inc defined below.


    ldx = 0;  %ldx is index for reduced factor vector, which is constructed by eliminating groups of redundant factors.


    inc = 2^(m-jdx);  %size of groups at level jdx

    Ntwo = max(size(factor_index_vector));    %length of factor vector, which will change as unimportant factors eliminated.


    no_groups = Ntwo/inc;  %number of groups for level jdx


    for idx = 1:no_groups    %for each of the groups of size inc


        test_stat = simulateD((idx-1)*inc, (idx)*inc, N, noreps, noreps);

        %Recall equation for expression of D in Cheng's paper after eq (8).
        % D is unbiased for the sum of factors from index j+1 to k. Hence need to subtract one here to match up code.


        thresh = B - tinv(1-test_size, N-1)*sqrt(var_est)*sqrt(2/noreps);

        %For one sided CI need to use t-statistic quantile as variance is estimated.
        % Here is the main difference to Cheng's approach, since the variance is estimated previously and not a pooled estimate.


        if test_stat > thresh     %There is at least one important factor in this group, so indices retained for next stage

            disp('Group containing important factors identified')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);

        else


            disp('Group containing unimportant factors eliminated')

        end



    end


    factor_index_vector = temp_factor_vec_clone;
    %Replace the factor index vector with the one with unimportant factor indices eliminated.
    temp_factor_vec_clone = 0;
    %Need to do this to initialise for next pass through the algorithm.




end


%Construct initial confidence intervals for further classification of
%factors.


for kdx=1:max(size(factor_index_vector))

    current_index = factor_index_vector(kdx);
    Dval = simulateD(current_index-1, current_index, N, noreps, noreps);

    flag = 0;

    while flag == 0
    low_int_limit = Dval - tinv(1-test_size, N-1)*sqrt(var_est)*sqrt(2/noreps);
    upp_int_limit = Dval + tinv(1-test_size, N-1)*sqrt(var_est)*sqrt(2/noreps);

    %This is Cheng's classification algorithm on page 278.
     %If the algorithm enters this stage it is necessary to increase the number of replications until the factor can be classified.

    if (B > low_int_limit) && (B < upp_int_limit)

        noreps = noreps + 10;
        Dval = simulateD(current_index-1, current_index, N, noreps, noreps);
        low_int_limit = Dval - tinv(1-test_size, N-1)*sqrt(var_est)*sqrt(2/noreps);
        upp_int_limit = Dval + tinv(1-test_size, N-1)*sqrt(var_est)*sqrt(2/noreps);

    else

        flag = 1;

    end

    end


    low_int_limit_vec(kdx) = low_int_limit;
    upp_int_limit_vec(kdx) = upp_int_limit;


end


disp('List of indices of important factors')

ldx=0;

for kdx = 1:max(size(factor_index_vector))


    if B < low_int_limit_vec(kdx)
         ldx = ldx + 1;
        imp_fac(ldx) = factor_index_vector(kdx);

    elseif  upp_int_limit_vec(kdx) < B + a

        ldx = ldx + 1;
        imp_fac(ldx) = factor_index_vector(kdx);

    end

end



imp_fac



low_int_limit_vec



upp_int_limit_vec


%The (1-\alpha) percent confidence intervals for important factors are
%(low_int_limit_vec(j), upp_int_limit_vec(j))) for the $j$th element of
%imp_fac.


end
```

A.2. Function simulateD

```
function f = simulateD(j, k, N, rk, rj);


%This code produces a realisation of the test statistic D in Cheng's paper,
%with replication to reduce variance.



f = sum(simulate_model(k, N, rk))/rk - sum(simulate_model(j, N, rj))/rj;
```

A.3. Function simulate_model

```
function f = simulate_model(j, N, M);

%simulates y^{(j)} in Cheng's paper, where the metamodel has the first j
```

```
%components switched on and the remaining N-j components switched off.
%This code requires the meta model test_meta_model.


for kdx=1:M

vec = [ones(1, j), zeros(1,N-j)];

yval(kdx) = test_meta_model(vec);


end


f = yval;
```

A.4. Function test_meta_model

```
%This is the generic meta model used for testing the SB algorithms. There
%are a number of alternatives available for testing the algorithm. The frst
%three are based upon a modification of the example in Bettonvil and
%Kleijnen. It is necessary to pad out with zeros to increase the factor
%vector length to a power of two.
%
%
%
%Most significant factors are at 14-18, and 20-24, with the largest factor
%at 20 for the first model.
%
%
%The model's variance can either be set as a constant of determined
%randomly in a given interval. This is good to test the model's resilience
%to non-uniform variance, which is a limitation of Cheng's method.


function out = test_meta_model( X )

%sigma = 1;

%generate sigma randomly in an interval [aval, bval]

aval = 1;
bval = 5;
sigma = aval + (bval-aval)*rand;

beta = [5.7, 0.6, 4.5, 19.3, 9.1, 28.4, 51.0, 34.6, 23.3, 39.7, 45.4, 93, 73.7, 144.6, 130.4, 42.5, 313.0, ...
166.2, 76.6, 345.4, 195.7, 188.8, 148.0, 206.4, 0, 0, 0, 0, 0, 0, 0, 0];



betazero = 3.2;


for i = 1:32


    y(i) = beta(i)*X(i);

end



out=  betazero + sum(y) + sigma*randn;

end
```

Appendix B. Sequential Bifurcation: Version I

B.1. Main Algorithm

```
%First version of controlled SB, based upon Wan, Ankenman and Nelson
%(2006). This code operates on the meta-model stored in the separate matlab
%file test_meta_model, which allows the user to test between a series of
%alternatives, such as the case where there are no important factors, all
%are important and a test case based upon Bettonvil and Kleijnen. It is
%necessary to have a meta_model with the number of factors a power of two.
%this can be achieved in other cases by padding out with zeros, provided
%the zeros are placed at the end of the meta model. It is interesting to
%note that the algorithm is unpredicatable if the zeros are placed in the
%initial list. This effect seems to have been reported by other
%investigators. The number of factors (N) must be specified here but the
%algorithm is blind to the meta-model. It is assumed that the meta-model
%can be simulated though, which is the standard practice in the
%literature. The algorithm attempts to estimate the B in the SB algorithm
%by estimating statistics from the meta-model. An indifference parameter
%(lim_par) must be specified by the user, as a tolerance.

clear all;

close all;



N = 32; %Number of factors


noreps = 100;
noreps2 = 100;

m = log2(N);  %power of two


factor_index_vector = 1:N;
%This vector stores the factor index of important factors as SB proceeds.
%Hence the jth entry of this is the index of the jth remaining factor as the algorithm proceeds.
%The length of this vector will change dynamically in the SB process.

for kdx=1:noreps2
    res1(kdx) = test_meta_model(zeros(1,N));
    res2(kdx) = test_meta_model(ones(1,N));
end

var_est = var(res1);

lim_par = 10;   %Selection of limit--user defined.

B = mean(res2)/noreps2 + lim_par*sqrt(var_est);
%Using an automatic approach for determination of thresholds.



delta0 = B;   %Threshold for factor importance.
delta1 = B+20; %Lower limit for power of test.
gamma = 0.95; %Power of test
alpha = 0.05; %Size of test



%The algorithm begins by assesssing whether there is at least one important
%factor. This is adopted from Cheng's approach since it is just a
%preliminary test to avoid the main algorithm executing on a set of
%unimportant factors. Without this the main algorithm messes up on a set of
%unimportant factors.

test_statN = simulateD(0, N, N, noreps, noreps);
test_stat_0 = sum(res1)/noreps2;   %This is matched to test_stat2 indirectly.


if test_statN - test_stat_0  <= B - tinv(1-alpha, N-1)*sqrt(var_est)*sqrt(2/noreps2);
%no important factors at all

   disp('No important factors');

else %there is at least one important factor






for jdx=1:m


    ldx = 0;
    %ldx is index for reduced factor vector, which is constructed by eliminating groups of redundant factors.
    inc = 2^(m-jdx);  %size of groups at level jdx

    Ntwo = max(size(factor_index_vector));
    %length of factor vector, which will change as unimportant factors eliminated.


    no_groups = Ntwo/inc;  %number of groups for level jdx


    for idx = 1:no_groups    %for each of the groups of size inc


        test_stat = simulateD((idx-1)*inc, (idx)*inc, N, noreps, noreps);
        %recall equation for exp of D in Cheng's paper after eq (8).
        % D is unbiased for the sum of factors from index j+1 to k. Hence need to subtract one here to match up code.

        threshU = UThresh((idx-1)*inc,(idx)*inc, N, delta0, alpha, noreps, noreps, noreps);
        threshL = LThresh((idx-1)*inc,(idx)*inc, N, delta0, gamma, noreps, noreps, noreps);


        %Begin the classification process for the current group of factors.

        if (test_stat <= threshU) && (noreps >= Nval((idx-1)*inc, (idx)*inc, N, noreps, alpha, gamma, delta0, delta1))

        disp('Group of unimportant factors eliminated');   %Factors not retained for next stage.




        elseif test_stat <= threshL

               disp('Group of unimportant factors eliminated');   %Factors not retained for next stage.



            elseif  test_stat > threshU
            %there is at least one important factor in this group, so indices retained for next stage


                disp('Group containing important factors identified: stage 1')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);



        else  %Exectute second stage of determining whether there are any important factors.

          Nstat = Nval((idx-1)*inc, (idx)*inc, N, noreps, alpha, gamma, delta0, delta1);
          noreps3 = max(Nstat, noreps);

          test_stat = simulateD((idx-1)*inc, (idx)*inc, N, noreps3, noreps3);
          threshU = UThresh((idx-1)*inc,(idx)*inc, N, delta0, alpha, noreps, noreps3, noreps3);
          %Second stage threshU, different noreps.

          if test_stat > threshU     %there is at least one important factor in this group, so indices retained for next stage

              disp('Group of important factors identified: stage 2')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);

          end


        end



    end


    factor_index_vector = temp_factor_vec_clone;
    temp_factor_vec_clone = 0;



end


disp('List of indices of important factors')

factor_index_vector


end
```

B.2. Function UThresh

```
function f = UThresh(i,j, N, delta0, alpha, n0, ni, nj)

f = delta0 + tinv(sqrt(1-alpha), n0-1)*sqrt(ssqr(i, j, N, n0))/sqrt(min(ni, nj));
```

B.3. Function LThresh

```
function f = LThresh(i,j, N, delta0, gamma, n0, ni, nj)

f = delta0 - tinv(0.5*(1+gamma), n0-1)*sqrt(ssqr(i, j, N, n0))/sqrt(min(ni, nj));
```

B.4. Function ssqr

```
function f = ssqr(i, j, N, n0);
for kdx=1:n0
Dval(kdx) = simulateD(i, j, N, 1, 1);

end

f = sum((Dval - mean(Dval)).^2)/(n0-1);
```

B.5. Function Nval

```
function f = Nval(i, j, N, n0, alpha, gamma, delta0, delta1);
f = ((h(alpha, n0, gamma))^2)*ssqr(i, j, N, n0)/(delta1-delta0)^2;
```

B.6. Function h

```
function f = h(alpha, n0, gamma)

f = tinv(sqrt(1-alpha), n0) - tinv((1-gamma)/2, (n0-1));
```

This page is intentionally blank

Appendix C. Sequential Bifurcation: Version II

C.1. Main Algorithm

```
%Second version of controlled SB, based upon Wan, Ankenman and Nelson
%(2010). This code operates on the meta-model stored in the separate matlab
%file test_meta_model, which allows the user to test between a series of
%alternatives, such as the case where there are no important factors, all
%are important and a test case based upon Bettonvil and Kleijnen. It is
%necessary to have a meta_model with the number of factors a power of two.
%This can be achieved in other cases by padding out with zeros, provided
%the zeros are placed at the end of the meta model. It is interesting to
%note that the algorithm is unpredicatable if the zeros are placed in the
%initial list. This effect seems to have been reported by other
%investigators. The number of factors (N) must be specified here but the
%algorithm is blind to the meta-model. It is assumed that the meta-model
%can be simulated though, which is the standard practice in the
%literature. The algorithm attempts to estimate the B in the SB algorithm
%by estimating statistics from the meta-model. An indifference parameter
%(lim_par) must be specified by the user, as a tolerance.

clear all;

close all;



N = 32; %Number of factors


noreps = 500;
noreps2 = 500;

m = log2(N);  %power of two

tic

factor_index_vector = 1:N;
%This vector stores the factor index of important factors as SB proceeds.
%Hence the jth entry of this is the index of the jth remaining factor as the algorithm proceeds.
 %The length of this vector will change dynamically in the SB process.

for kdx=1:noreps2
    res1(kdx) = test_meta_model(zeros(1,N));
    res2(kdx) = test_meta_model(ones(1,N));
end

var_est = var(res1);

lim_par = 5;   %Selection of limit--user defined.

B = mean(res2)/noreps2 + lim_par*sqrt(var_est);
%Using an automatic approach for determination of thresholds.



delta0 = B;   %Threshold for factor importance.
delta1 = B+20; %Lower limit for power of test.
%%%
%%%
%%% For this situation it is assumed that alpha+gamma = 1
%%%
%%%

gamma = 0.95; %Power of test
alpha = 0.05; %Size of test
rzero = (delta0+delta1)/2;
azero = (exp( (-2*log(2*alpha))/(noreps-1)) - 1)*(noreps-1)/(delta1-delta0);
lambval = (delta1-delta0)/4;

%The algorithm begins by assesssing whether there is at least one important
%factor. This is adopted from Cheng's approach since it is just a
%preliminary test to avoid the main algorithm executing on a set of
%unimportant factors. Without this the main algorithm messes up on a set of
%unimportant factors.


test_statN = simulateD(0, N, N, noreps, noreps);
test_stat_0 = sum(res1)/noreps2;   %This is matched to test_stat2 indirectly.


if test_statN - test_stat_0  <= B - tinv(1-alpha, N-1)*sqrt(var_est)*sqrt(2/noreps2);
%no important factors at all

   disp('No important factors');

else %there is at least one important factor
```

```
for jdx=1:m


    ldx = 0;
    %ldx is index for reduced factor vector, which is constructed by eliminating groups of redundant factors.


    inc = 2^(m-jdx);
    %size of groups at level jdx

    Ntwo = max(size(factor_index_vector));
    %length of factor vector, which will change as unimportant factors eliminated.


    no_groups = Ntwo/inc;
    %number of groups for level jdx

    rval = noreps;

    for idx = 1:no_groups
    %for each of the groups of size inc



        test_stat = rval*(simulateD((idx-1)*inc, (idx)*inc, N, rval, rval) - rzero);
        %recall equation for exp of D in Cheng's paper after eq (8).
        % D is unbiased for the sum of factors from index j+1 to k. Hence need to subtract one here to match up code.




        %Begin the classification process for the current group of factors.

        flag = sign(rzero -  Mval((idx-1)*inc, (idx)*inc, alpha, delta0, delta1, N, noreps));


        switch flag;

           case +1  %rzero > Mval((idx-1)*inc, (idx)*inc, delta0, delta1, N, noreps)


                       if test_stat <= 0


                              disp('Group of unimportant factors eliminated');
                              %Factors not retained for next stage.



                       else

           disp('Group containing important factors identified: stage 1')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);

                       end



            case -1   %(rzero <= Mval((idx-1)*inc, (idx)*inc, delta0, delta1, N, noreps))




            if test_stat <= -azero*ssqr((idx-1)*inc, (idx)*inc, N, rval) + lambval*rval

               disp('Group of unimportant factors eliminated: stage 2');
               %Factors not retained for next stage.


            elseif  test_stat >= azero*ssqr((idx-1)*inc, (idx)*inc, N, rval) - lambval*rval
            %there is at least one important factor in this group, so indices retained for next stage


                disp('Group containing important factors identified: stage 2')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);
            else

          rval = rval + 1;


            end

    end



    end






    factor_index_vector = temp_factor_vec_clone;
    temp_factor_vec_clone = 0;



end


toc

disp('List of indices of important factors')

factor_index_vector

end
```

This page is intentionally blank

Appendix D. Fold Over Algorithm for Interactions

D.1. Main Algorithm

```
%This version modified to run in the case of interactions. The algorithm is
%the fold-over design in Wan et al (2010). This algorithm eliminates the
%second order interactions to allow for testing of the meta-model's main
%effects. It is designed to operate with the meta-model specified in test_meta_model_interactions


clear all;

close all;



N = 32; %Number of factors


noreps = 1000;
noreps2 = 1000;

m = log2(N);  %power of two

tic

factor_index_vector = 1:N;
%This vector stores the factor index of important factors as SB proceeds.
%Hence the jth entry of this is the index of the jth remaining factor as the algorithm proceeds.
 %The length of this vector will change dynamically in the SB process.

for kdx=1:noreps2
    res1(kdx) = test_meta_model_interactions(zeros(1,N));
    res2(kdx) = test_meta_model_interactions(ones(1,N));
end

var_est = var(res1);

lim_par = 5;   %Selection of limit--user defined.

B = mean(res2)/noreps2 + lim_par*sqrt(var_est);
%Using an automatic approach for determination of thresholds. Note that res2 will be biased by the interactions. This is difficult to get around.



delta0 = B;   %Threshold for factor importance.
delta1 = B+20; %Lower limit for power of test.
%%%
%%%
%%% For this situation it is assumed that alpha+gamma = 1
%%%
%%%

gamma = 0.95; %Power of test
alpha = 0.05; %Size of test
rzero = (delta0+delta1)/2;
azero = (exp( (-2*log(2*alpha))/(noreps-1)) - 1)*(noreps-1)/(delta1-delta0);
lambval = (delta1-delta0)/4;

%The algorithm begins by assesssing whether there is at least one important
%factor. This is adopted from Cheng's approach since it is just a
%preliminary test to avoid the main algorithm executing on a set of
%unimportant factors. Without this the main algorithm messes up on a set of
%unimportant factors.

test_statN = simulateDI(0, N, N, noreps, noreps);
test_stat_0 = sum(res1)/noreps2;   %This is matched to test_stat2 indirectly.


if test_statN - test_stat_0  <= B - tinv(1-alpha, N-1)*sqrt(var_est)*sqrt(2/noreps2);
%no important factors at all

   disp('No important factors');

else %there is at least one important factor






for jdx=1:m


    ldx = 0;
    %ldx is index for reduced factor vector, which is constructed by eliminating groups of redundant factors.


    inc = 2^(m-jdx);
    %size of groups at level jdx
    Ntwo = max(size(factor_index_vector));
    %length of factor vector, which will change as unimportant factors eliminated.


    no_groups = Ntwo/inc;
    %number of groups for level jdx

    rval = noreps;

    for idx = 1:no_groups
    %for each of the groups of size inc



        test_stat = rval*(simulateDI((idx-1)*inc, (idx)*inc, N, rval, rval) - rzero);
        %recall equation for exp of D in Cheng's paper after eq (8).
        %D is unbiased for the sum of factors from index j+1 to k.
        %Hence need to subtract one here to match up code.




        %Begin the classification process for the current group of factors.

        flag = sign(rzero -  Mval((idx-1)*inc, (idx)*inc, alpha, delta0, delta1, N, noreps));



        switch flag;

           case +1  %rzero > Mval((idx-1)*inc, (idx)*inc, delta0, delta1, N, noreps)



                       if test_stat <= 0


                              disp('Group of unimportant factors eliminated');
                              %Factors not retained for next stage.



                       else

           disp('Group containing important factors identified: stage 1')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);

                       end




            case -1   %(rzero <= Mval((idx-1)*inc, (idx)*inc, delta0, delta1, N, noreps))




            if test_stat <= -azero*ssqrI((idx-1)*inc, (idx)*inc, N, rval) + lambval*rval

               disp('Group of unimportant factors eliminated: stage 2');
               %Factors not retained for next stage.



            elseif  test_stat >= azero*ssqrI((idx-1)*inc, (idx)*inc, N, rval) - lambval*rval
            %there is at least one important factor in this group, so indices retained for next stage


                disp('Group containing important factors identified: stage 2')

            ldx = ldx + 1;    %An index for the vector of important factors

                len = max(size(((idx-1)*inc+1):(idx)*inc));

                temp_factor_vec_clone((1 + (ldx-1)*len):ldx*(len)) = factor_index_vector(((idx-1)*inc + 1):(idx)*inc);




            else

          rval = rval + 1;


            end

    end
    end



    factor_index_vector = temp_factor_vec_clone;
    temp_factor_vec_clone = 0;


end

toc

disp('List of indices of important main factors')

factor_index_vector

end
```

D.2. Function Mval

```
function f = Mval(i, j, alpha, delta0, delta1, N, n0);


a0 = (exp( (-2*log(2*alpha))/(n0-1)) - 1)*(n0-1)/(delta1-delta0);

aval = ssqrI(i, j, N, n0)*a0;

lambval = (delta1-delta0)/4;

f = floor(aval/lambval);
```

D.3. Function simulate mirror model

```
function f = simulate_mirror_model(j, N, M);


for kdx=1:M

vec1 = [ones(1, j), zeros(1,N-j)];
vec2 = [-1*ones(1, j), zeros(1,N-j)];

yval(kdx) = 0.5*(test_meta_model_interactions(vec1)-test_meta_model_interactions(vec2));


end


f = yval;
```

D.4. Function simulateDI

```
function f = simulateDI(j, k, N, rk, rj);


%This code produces a realisation of the test statistic D in Cheng's paper,
%with replication to reduce variance.


f = sum(simulate_mirror_model(k, N, rk))/rk - sum(simulate_mirror_model(j, N, rj))/rj;
```

D.5. Function ssqrI

```
function f = ssqrI(i, j, N, n0);

%i = 2; j = 8; N=32; n0=10
for kdx=1:n0

Dval(kdx) = simulateDI(i, j, N, 1, 1);


end


f = sum((Dval - mean(Dval)).^2)/(n0-1);
```

D.6. Function test_meta_model_interactions

```
function out = test_meta_model_interactions( X )

%sigma = 1;

%generate sigma randomly in an interval [aval, bval]

aval = 1;
bval = 5;
sigma = aval + (bval-aval)*rand;

interactionLevel1 = 10;
interactionLevel2 = 20;

beta = [5.7, 0.6, 4.5, 19.3, 9.1, 28.4, 51.0, 34.6, 23.3, 39.7, 45.4, 93, 73.7, 144.6, 130.4, 42.5, 313.0, 166.2, 76.6, 345.4, ...
195.7, 188.8, 148.0, 206.4, 0, 0, 0, 0, 0, 0, 0, 0];


betazero = 3.2;


for i = 1:32


    y1(i) = beta(i)*X(i);



end

%%%%It is interesting to examine the effect of increasing interaction
%%%%strength on the classification of the main effects model. Simulations
%%%%show that if the interaction is not too stong, most significant main
%%%%effects seem to be correctly classified. When the interaction strength
%%%%is increased, some important main effects get misclassified.

y2(1)=0;

for i = 2:32

    y2(i) = interactionLevel1*X(i-1)*X(i)  + interactionLevel2*X(i)*X(i);


end


%y2(32) = 0;


for i=1:32


     y(i) = y1(i) + y2(i);


end


out=  betazero + sum(y) + sigma*randn;

end
```

DEFENCE SCIENCE AND TECHNOLOGY GROUP DOCUMENT CONTROL DATA	1. DLM/CAVEAT (OF DOCUMENT)
2. TITLE Factor Screening Techniques for Combat Simulation Models	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED LIMITED RELEASE USE (U/L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR Graham V. Weinberg	5. CORPORATE AUTHOR Defence Science and Technology Group 506 Lorimer St, Fishermans Bend, Victoria 3207, Australia

6a. DST GROUP NUMBER DST-Group–TN–1919	6b. AR NUMBER	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE September, 2019

8. OBJECTIVE ID	9. TASK NUMBER	10. TASK SPONSOR

11. MSTC Land Capability Analysis	12. STC Combined Arms Simulation
13. DOWNGRADING/DELIMITING INSTRUCTIONS `http://dspace.dsto.defence.gov.au/dspace/`	14. RELEASE AUTHORITY Chief, Joint and Operations Analysis Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

Approved for public release.

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. RESEARCH LIBRARY THESAURUS

statistical analysis, statistics, numerical algorithms, computer applications

19. ABSTRACT

A stochastic simulation model can be viewed as a system that takes a series of input variables, and then processes them to produce a realisation of the required simulation scenario output. This model will contain a set of parameters, known as factors, that characterise the simulation process. There may in fact be significant complexity with the underlying simulation architecture, and additionally there may be a very large number of factors. The cost of this is slow performance in simulation run times. In many cases there may be factors in the underlying model that are insignificant statistically in the scenario of interest. This phenomenon is referred to as the parsimony principle, or the Pareto 80-20 rule. Hence there is much interest in the scientific literature on the screening of factors in the underlying meta-model applied to the dynamical system. This report thus investigates the subject of factor screening, for stochastic simulation models, and overviews several solutions to this problem. In particular, sequential bifurcation will be shown to provide a very efficient approach to the problem of factor screening, in comparison to standard one factor at a time classification methods. Several variations of sequential bifurcation will be examined, and their performance in several factor screening examples will be investigated. Directions for possible future research will also be discussed.