ROBUST OPTIMIZATION THROUGH RESPONSE SURFACE METHODOLOGY

(RESPONSE SURFACE KULLANAN SAĞLAMCI ENİYİLEME)

by

Seda EYİGÜN, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Date of Submission : May 4, 2009

Date of Defense Examination: June 3, 2009

Supervisors : Asst. Prof. Dr. M. Ebru ANGÜN
Assoc. Prof. Dr. Esra ALBAYRAK

Committee Members: Assoc. Prof. Dr. Temel ÖNCAN
Assoc. Prof. Dr. Mehmet Mutlu YENİSEY
Asst. Prof. Dr. Cafer Erhan BOZDAĞ

ACKNOWLEDGEMENTS

I would like to thank Yrd. Doç. Dr. M. Ebru Angün and Doç. Dr. Esra Albayrak, for their guidance and support throughout the preparation of this study despite all difficult conditions around us.

I would also like to thank Rengin Burhanoğlu for her friendship of the past ten years, her permanent desire to be my colleague in every project at Galatasaray University and just for being herself any time.

I would also like to extend special thanks to my manager at work; Ayşe Yenidoğan who provided me with the time I needed to finalize this thesis, and who showed an endless patience for my mind divided in three for the past six months.

Finally, I would like to thank my family for their love and trust in me and my husband Yılmaz Cem Şenol.

SEDA EYİGÜN
JUNE 2009

TABLE OF CONTENTS

LIST OF SYMBOLS

DRS: Dual Response Surface

RSM: Response Surface Methodology

RPD: Robust Parameter Design

CCP: Chance Constrained Programming

CCD: Central Composite Design

OAs: Orthogonal Arrays

SN: Signal-to-noise

OLS: Ordinary Least Squares

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

During the last couple of decades, optimization methods have become one of the most important research fields and received increasing attention from engineers, product designers and researchers especially in the field of production and related industries.

The classical optimization approaches assume that the input data of the optimization model is known with certainty. However, in many areas of application of real world problems like inventory management, portfolio selection, supply chain optimization and production planning, it's needed to integrate the uncertainty of the input data into the optimization model which refers to optimization under uncertainty.

The increasing interest in simulation optimization for problems that arise in practical applications becomes relevant where explicit mathematical formulations are too restrictive. Therefore, for many practical cases one cannot obtain an analytical solution through those kind of methods. Indeed, simulation optimization has led to the numerical solution of large-scale, real-world decision-making problems.

A simulation optimization method, Response Surface Methodology (RSM); aims to achieve optimum operation conditions while minimizing the variability in order to produce high quality and reliable products and services at the lowest possible cost. As an extension of Robust Parameter Design (RPD), RSM is a combination methodology of mathematical and statistical techniques in problem modeling and analysis.

The risk-neutrality problem of the classic simulation optimization problems can be handled by the Dual Response Surface (DRS) approach within RSM and combining it with the Taguchi's RPD enables researchers to cope with the *unknown* environments.

In this work, a risk-averse approach to Response Surface Methodology, which explicitly deals with random environments, is presented. The main contribution of this thesis is to adapt Taguchian RSM to discrete-event simulation studies.

The thesis introduced the steps of this Taguchian RSM approach and then an application of these steps to an inventory optimization is provided. The computer program is coded in *Matlab 7.6.* , and the optimization is performed through the built-in function *fmincon* in *Matlab*.

Furthermore, Taguchian RSM method is applied to a more complex example which is a call center problem modeled in *Arena*. The results taken from the execution of the model is used in our optimization algorithm coded in *Matlab*.

Although it's usefull to increase the number of decision variables in the example, because of the version limits of *Arena*, an example with an additional environmental factor is provided in order to expand the original example. Thus we left this issue as a future research.

For the future work, this study can be extended to an iterative approach, or the proposed approach can be developed to handle multiple random responses.

RESUME

Pendant les dernières décades, les méthodes d'optimisation sont devenues une des régions de recherche les plus importantes et ont reçus l'attention augmentante des ingénieurs, des créateurs de produit et des chercheurs surtout dans le domaine de la production.

Les approches d'optimisation classiques supposent que les variables d'entrée du modèle d'optimisation sont connues avec la certitude. Cependant, dans le monde réel, il est nécessaire d'intégrer l'incertitude des variables d'entrée dans le modèle d'optimisation qui se réfère à l'optimisation sous l'incertitude.

L'intérêt croissant sur l'optimisation de simulation pour les problèmes qui se présentent dans les applications réel devient pertinentes où les formulations mathématiques sont trop restrictives. Donc, pour ces type de problèmes, l'un ne peut pas obtenir une solution analytique par ces type de méthodes. En effet, l'optimisation de simulation a mené à la solution numérique d'aux problèmes de prise de décision à grande échelle.

Une méthode d'optimisation de simulation; La Méthodologie de Surface de Réponse (RSM) a l'intention d'accomplir des conditions d'opération optimales, en minimisant la variabilité, pour produire des produits et des services de haute qualité au prix le plus bas possible. Comme une extension de Plan Paramètre Robuste (RPD), RSM est une méthodologie de combinaison de techniques mathématiques et statistiques, utilisée pour le modèle at l'analyse du problème.

Le problème de risque-neutralité des problèmes classiques d'optimisation de simulation peut être traité par l'approche Dual Response Surface (DRS) dans RSM et la

combinaison de cette approache avec le Plan Paramètre Robuste (RPD) du Taguchi permet à chercheurs de faire face aux environnements inconnus.

Dans ce travail, une approche risque-opposé à la Méthodologie de Surface de Réponse, qui traite explicitement des environnements faits au hasard, est présentée. La contribution principale de cette thèse est d'adapter Taguchian RSM aux études de simulation de discret-événement.

L'étude introduit les étapes de cette approach et une application de ces étapes à un problème d'optimisation d'inventaire est fourni. Le programme informatique est codé dans *Matlab 7.6.* , et l'optimisation est exécutée par le *fmincon* de fonction intégré dans *Matlab*.

En outre, la méthode de Taguchian RSM est appliquée à une exemple plus complexe qui est un problème de centre téléphonique et est modelé dans *Arena*. Les résultats pris de l'exécution du modèle sont utilisés dans notre algorithme d'optimisation, codé dans *Matlab*.

Bien que c'est utile d'augmenter le nombre de variables de décision dans l'exemple, à cause des limites de version d'*Arena*, un exemple avec un facteur ambiant supplémentaire est fourni pour grandir l'exemple original. Ainsi nous sommes partis ce problème comme une recherche future.

Pour le travail futur, cette étude peut être étendue à une approche itérative, ou l'approche proposée peut être développée pour manipuler des réponses multiples faites au hasard.

ÖZET

Optimizasyon metodları özellikle üretim ve ilgili endüstrilerde, birçok mühendis, tasarımcı ve araştırmacı tarafından kullanılan önemli çalışma alanlarından birisidir.

Klasik yaklaşımların çoğunda girdiler bilinir durumda olsa da, gerçek hayattaki birçok problemde (envanter yönetimi, portfolio seçim, tedarik zinciri ve üretim planlama problemleri gibi) girdi verilerinin bilinir olmaması, araştırmacıları farklı optimizasyon tekniklerinin geliştirilmesine yönlendirmiştir.

Gerçek hayattaki birçok problemin çözüm yaklaşımında kısıt ve geçerlilikleri kesin olan matematiksel formüllerin kullanımı sınırlayıcı olduğundan, bu tip problemlerde simülasyon-optimizasyon metodlarının kullanılması daha geçerli hale gelmektedir. Bu tip problemlerde matematiksel formüller kullanarak analitik bir çözüm elde etmek oldukça zordur. Bu yüzden, simülasyon-optimizasyon yaklaşımı gerçek hayatta karşılaşılan, büyük ölçekli karar verme problemlerinde sayısal sonuçlar için yol gösterici olabilmektedir.

Düşük maliyetli, yüksek kaliteli ve güvenilir ürünler üretmek ya da bu ölçülerde hizmet sağlamak amacıyla uygulanan Sağlamcı Parametre Tasarımı (Robust Parameter Design - RPD) tekniği , değişkenliği en aza indirerek optimum operasyon koşullarını elde etmeyi amaçlamaktadır.

Tepki Yüzeyi Metodolojisi (Response Surface Methodology - RSM), Sağlamcı Parametre Tasarımı tekniğinin bir uzantısı ve bir simülasyon-optimizasyon metodu olarak, problem analizi ve modellemede matematiksel ve istatistiksel tekniklerin biraraya gelmesiyle oluşturulmuş bir yöntemdir.

Simülasyon optimizasyon problemlerinde kullanılan klasik yaklaşımların risk-nötr olması problemi ve bu problemlerin raslantısal çevrelerde çözümlenebilmeleri çift tepki yüzeyi yaklaşımı ve bu yaklaşımın Taguchi'nin Sağlamcı Parametre tasarımı ile birleştirilmesi ile çözümlenebilmektedir.

Bu çalışmada, Tepki Yüzeyi Metodolojisi yöntemi; raslantısal çevrelerde riskten kaçınma yöntemi ile incelenerek, yeni bir yaklaşım önerilmiştir. Çalışmanın temel katkısı, önerilen yöntemin ayrık olaylı simülasyon örneklerine uygulanmasıdır.

Çalışmanın genelinde, bu yaklaşımın adımları belirtilmiş ve bu adımlar bir envanter optimizasyon problemine uygulanarak sonuçları analiz edilmiştir. İlgili bilgisayar programı Matlab 7.6.'da yazılmış, optimizasyon da Matlab'ın *fmincon* fonksiyonu ile uygulanmıştır.

Daha sonra, sözkonusu metod daha kompleks bir örnek olan ve Arena programı üzerinde modellenen çağrı merkezi problemine iki farklı durumda uygulanmıştır. Modelin çalışması sonucu elde edilen sonuçlar Matlab'da yazılan optimizasyon algoritmasında kullanılmıştır.

Örnekteki karar değişkelerinin sayısını arttırmak uygulama için faydalı olacaktır. Ancak simülasyon çalışması için kullanılan Arena programının öğrenci versiyonu, bu artırımı kısıtladığından, örneği geliştirmek için çevresel faktör eklenmiştir. Bu durum, ileriki bir uygulama olarak bırakılmıştır.

Çalışma sonucunda, önerilen bir kerelik yaklaşımın yinelemeli bir yaklaşıma dönüştürülebileceği, ya da çoklu raslantısal tepki problemleri için geliştirilebileceği belirtilmiştir.

1. INTRODUCTION

In this chapter, we will present an overview on different approaches to optimization under uncertainty, including two-stage stochastic programming with recourse, probabilistic (or chance constraint) programming, risk-averse optimization and robust optimization; reviews for stochastic programming with recourse, probabilistic programming, and robust optimization are given in recent papers of Sahinidis [1] and Beyer and Sendhoff [2]. Risk-averse optimization is explained in detail in Shapiro and Ruszczyński's comprehensive survey about stochastic programming [3].

For real-world optimization problems, the decision environment is usually characterized by the following facts:

- The parameters (e.g., cost vector) are estimated through historical data. Hence, they are uncertain.
- The optimal solution, even if computed very accurately, may be difficult to be implemented accurately.
- The problem must remain feasible for all meaningful realizations of the parameters.
- Problems are large-scale. There are in general many variables and/or constraints.
- Bad optimal solutions (those that become severely infeasible when the parameters of the problem are slightly changed) are quite common.

These facts imply that in many cases we deal with optimization problems under uncertainty; see also many application papers on, for example, inventory management, portfolio selection, facility planning, supply chain optimization, and production planning and scheduling. The classical approaches to linear and nonlinear optimization problems, on the other hand, assume that the parameters of the optimization problem are known with certainty. Therefore, it is important to present the methodologies that can

cope with optimization problems under uncertainty, as well as their advantages and shortcomings.

1.1. TWO STAGE STOCHASTIC PROGRAMMING WITH RECOURSE:

In the standard two-stage stochastic programming, the decision variables are partitioned into two sets. The first-stage variables are those that have to be decided before the actual realizations of the random parameters occur. Subsequently, once the realizations of the random parameters are obtained, the second-stage variables are determined as corrective measures or recourse against any infeasibilities arising due to these particular realizations of the random parameters at certain costs. Due to uncertainty, the second-stage cost is a random variable. Therefore, the objective is to select the first-stage variables such that the sum of the first-stage costs and the expected value of the random second-stage costs is minimized.

A standard formulation of the two-stage stochastic linear programming problem is as follows; for further information, standard textbooks on stochastic programming such as Kall and Wallace's, Birge and Louveaux's, and Shapiro and Ruszczyński's can be investigated [4, 5, 3].

$$\begin{aligned} &\text{minimize } c^T x + \mathrm{E}\big[Q(x,w)\big] \\ &\text{subject to } x \in X \end{aligned}$$

with (1.1)

$$\begin{aligned} Q(x,w) = \text{minimize } & f(w)^T y \\ \text{subject to } & D(w)y \geq h(w) + T(w)x \\ & y \in Y \end{aligned}$$

c is the cost vector for the vector x of the first-stage variables, X and Y are polyhedral sets, w is the vector of random variables from a probability space, $f(w)$ is the random cost vector for the vector y of the second-stage variables whose values depend on x, and $D(w)$, $T(w)$, and $h(w)$ are, respectively, random matrices and

random right-hand-side of the second-stage problem. The concept of recourse has been applied also to integer and non-linear programming, and to problems with multi-stages.

The main advantage of the two-stage stochastic linear programming problem is that under the assumption that w has a joint discrete distribution, the problem can be equivalently formulated as a large-scale linear programming problem which can be solved using standard linear programming technology. On the other hand, the main shortcoming of this approach is that infeasibilities at the second-stage are allowed at a certain penalty. The approach thus focusses on the minimization of the expected recourse costs without taking into account the system's reliability.

There have been many successful applications of stochastic programming in very diverse areas such as fleet assignment by Ferguson and Dantzig [6]; production of heating oil with constraints on demands and capacities by Charnes and Cooper [7]; water management systems by Dupačová, Gairovonski, Kos and Szantai [8]; energy planning by Manne; Louveaux; Pereira and Pinto; Manne and Richels; Morton; Takriti Birge and Long; Carøe, Ruszczyński, and Schultz [9, 10, 11, 12, 13, 14, 15]; forestry planning by Gassmann [16]; hospital staffing by Kao and Queyranne [17]; financial decision-making by Mulvey and Vladimirou; Ziemba and Vickson; Kallberg, White and Ziemba; Zenios; Dert; Carino and Ziemba; Kouwenberg and Zenios [18, 19, 20, 21, 22, 23, 24, 25, 26]; and capacity expansion problems by Sherali, Soyster, Murphy and Sen; Davis, Dempster, Sethi and Vermes; Bienstock and Shapiro; Eppen, Martin and Schrage; Berman, Ganz and Wagner; Malcom and Zenios; Ahmet, King and Parija [27, 28, 29, 30, 31, 32, 33].

1.2. CHANCE CONSTRAINT OR PROBABILISTIC PROGRAMMING:

In the recourse-based approach, decision-makers assign costs (penalties) to recourse activities that are taken to ensure feasibility of the second-stage problem. The focus is on the minimization of the expected recourse costs. In the probabilistic or chance constraint programming, however, the focus is on the reliability of the system; that is, the system's reliability to meet feasibility constraints in a random environment. This

reliability is expressed through one or many probability functions, which require that constraints are satisfied at a prespecified level.

Consider the following classical linear programming problem:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax \geq b \\ &\qquad x \geq 0 \end{aligned} \tag{1.2}$$

where c is the cost vector, x is the vector of decision variables, b is the right-hand-side vector, and A is the constraint matrix. Suppose that some entries in A are random and the constraints $Ax \geq b$ have to be satisfied with some probability $p \in (0,1)$. Now, the corresponding probabilistic programming problem can be given as [4, 5, 3]:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } P(Ax \geq b) \geq p \\ &\qquad \text{x} \geq 0 \end{aligned} \tag{1.3}$$

Suppose that in (1.3), there is only one constraint (hence, we have $P(a^T x \geq b) \geq p$) and randomness occurs in b. Suppose also that F is the cumulative density function of b. Then the problem with a single probabilistic constraint becomes a simple linear programming problem after replacing $P(a)$.

The main advantage of probabilistic programming is that it replaces the subjective penalties in the recourse-based approaches by probabilities. However, this objectivity has a price. In general, the feasible area of (1.3) is not convex, which makes (1.3) very difficult to be solved. The feasible set in (1.3) is convex only under restrictive assumptions [4, 5, 3].

Charnes and Cooper [34, 35] first introduced chance-constrained formulation of stochastic programs. The classic book by Vajda provides an excellent introduction to the formulation as well as various interpretations [36].

Applications of chance constraint programming (CCP) to capital rationing problems can be found in [37, 38]. An extensive review of stochastic investment planning is presented by Kelle, and Sarper [39, 40].

1.3. RISK AVERSE OPTIMIZATION:

In the following, we will explain risk averse optimization using the following simplest form of the newsvendor problem taken from Sylver, Pyke, and Peterson [41].

A newsvendor orders a fixed quantity x of newspapers to be sold each day. The daily demand D is assumed to be random, and the ordering decision should be made before a realization d of demand occurs. The per unit acquisition cost is c. Unsold newspapers are salvaged each day at the unit price w. A back order penalty cost of b per unit is incurred if d exceeds x. The question is to find an ordering quantity x that optimizes a selected performance measure, for example, the total cost. For a particular realization d, this total cost function can be formulated as

$$G(x,d) = cx + b(d-x)_+ - w(x-d)_+ \tag{1.4}$$

where $(d-x)_+$ and $(x-d)_+$ correspond to the maximum of $d-x$ and 0, and $x-d$ and 0, respectively.

If it makes sense to assume that the distribution function of D can be estimated from historical data, then one of the possible ways to formulate the newsvendor problem is to minimize the expected total cost, where the expectation is taken with respect to the distribution function of D:

$$\underset{x\geq 0}{\text{minimize}}\ \mathrm{E}\left[G\left(x,D\right)\right] \tag{1.5}$$

This classic formulation causes two well-known problems: First, it minimizes the total cost on average, and hence it does not take the decision-maker's attitude toward risk into account. Second, it is almost impossible to quantify the penalty cost b in $G\left(x,d\right)$.

These two problems can be overcome by the following chance constraint formulation of the newsvendor problem:

$$\begin{aligned}&\underset{x\geq 0}{\text{minimize}}\ \mathrm{E}\left[H\left(x,D\right)\right]\\&\text{subject to } P\left\{D-x>\tau\right\}\leq\alpha\end{aligned} \tag{1.6}$$

where for a particular realization d, $H\left(x,d\right)$ is the difference between the total acquisition cost and the revenue from the salvaged newspapers, if there are any, (that is, $H\left(x,d\right)=cx-w\left(x-d\right)_{+}$, and $P\left\{D-x>\tau\right\}\leq\alpha$ is the so-called probabilistic (or chance) constraint, which means that the probability of the demand exceeding the ordering quantity x by a predetermined threshold τ should not be greater than a predetermined significance level $\alpha\in\left(0,1\right)$. Hence, this approach minimizes a form of the cost function on average while making sure that the risk of the demand being larger than the ordering quantity is small. The problem type in (1.6) can be solved only after finding a deterministic equivalent of the probabilistic constraint. As we already mentioned in the previous subsection, the main disadvantages of having such a constraint is that its deterministic equivalent gives rise to a convex feasible set for the decision variable x (or vector in multi-dimensional case) only under restrictive assumptions on the distribution function of D. Otherwise, one has to deal with a nonconvex optimization problem.

Nemirovski and Shapiro construct convex approximations to probabilistic constraints. These approximations are conservative in the sense that the feasible sets defined by

these approximations are contained in the feasible sets defined by the probabilistic constraints [42]. Denoting the convex approximation of the probabilistic constraint by $\rho_{1-\alpha}(x, D)$, the problem in (1.6) becomes

$$\begin{aligned} &\text{minimize } \mathrm{E}\left[H(x, D)\right] \\ &\text{subject to } \rho_{1-\alpha}(x, D) \leq \alpha \end{aligned} \tag{1.7}$$

The problem type in (1.7) is called risk averse optimization in the stochastic programming literature [3]. As in (1.6), the problem in (1.7) minimizes the cost on average while reducing the risk of having more than τ backordered items to an acceptable level, namely $1-\alpha$. Some more properties of the problem in (1.7) are: (i) It is a convex optimization problem, and hence it is easily solvable by any optimization software; (ii) since the feasible set of (1.7) is contained in the feasible set of (1.6) and both problems have the same objective function, the minimum objective value of (1.7) provides an upper bound for the minimum objective value of (1.6).

This $\rho_{1-\alpha}$ has to satisfy some mathematical conditions which can be found in Artzner, Delbaen, Eber and Heath [43]. Furthermore, classic risk measures such as variance and standard deviation introduced in a portfolio selection problem by Markowitz do not satisfy some of these conditions [44, 45].

Risk averse optimization has applications in many fields; for example, Ahmed, Cakmak, and Shapiro apply this approach for inventory models [46], Rockafellar and Uryasev for portfolio optimization [47], and Garcia-Gonzalez, Parrilla, and Mateo for profit-based optimal scheduling of a hydro-chain [48].

Finally, in minimization problems, the risk aversion has been classically dealt with through disutility functions. The existence of such functions is derived axiomatically in Von Neumann and Morgenstern [49], but these disutility functions are very difficult to elicit in practice. With the risk averse approach, this problem disappears.

The main advantage of risk averse optimization is that it can be easily solved by standard convex optimization softwares. However, there are many different types for $\rho_{1-\alpha}$ such as semi-deviations and conditional-value-at-risk, and the choice for one of them introduces subjectivity to the problem.

1.4. ROBUST OPTIMIZATION:

Robust optimization is a modeling methodology, combined with computational tools, to process optimization problems in which the data are uncertain and is only known to belong to some uncertainty set.

There are two different approaches to robust optimization problems. The first approach is originated from Soyster [50], and further popularized by Ben-Tal and Nemirovski [51]. In this approach, the focus is on feasibility uncertainties; that is, uncertainties concerning the fulfillment of constraints the design variables must obey. This approach assumes certain types for the uncertainty sets and obtains a computationally tractable robust counterpart of the original problem. This approach also assumes that mathematical expressions for the objective and/or constraint functions are available, which is not the case for our problem. Therefore, in the rest of this work, we will not consider Ben-Tal and Nemirovski 's approach to robust optimization.

The second approach is originated from Taguchi. The main difference of Taguchi's method compared to ordinary optimization lies in the accounting for performance variations due to noise factors beyond the control of the designer. That is, there are two kinds of parameters entering the objective and/or constraint functions: control parameters which are to be tuned to optimality, and noise factors (e.g., environmental conditions) which are difficult to be controled by the designer [52].

Taguchi does not really use an automated optimization procedure. Instead, he uses design of experiments in order to evaluate different design (control) parameters. To this end, the design parameters are systematically changed taking values on a predefined (orthogonal) lattice, the so-called inner array. At each design point, the noise variables

are systematically changed according to an outer array. The outputs of the performance measures are obtained through real-life experimentation. Consequently, a statistical data analysis can be performed to identify the design variable producing the best performance.

From viewpoint of optimization efficiency, Taguchi's optimization approach suffers from the curse of dimensions. Suppose that we have a *k*-dimensional design vector and *g*-dimensional noise vector. Then, considering only the design vector, we already need 2^k experiments (either real-life or simulation runs). Adding also the noise vector, we will need a minimum of $g2^k$ experiments. As pointed out by Trosset: "the Taguchi approach violates a fundamental tenet of numerical optimization-that one should avoid doing too much work until one nears a solution" [53]. Besides these efficiency considerations, there are other aspects of Taguchi's method which are subject to controversial debates summarized in a panel discussion [54].

In our approach, we will use the idea of partitioning parameters into two sets, namely design and noise parameters. However, like Myers and Montgomery [55], and Dellino, Kleijnen, and Meloni [56], we will use Response Surface Methodology (RSM), which is a black box simulation optimization technique. RSM is a computationally efficient technique, which can cope with the curse of dimensions problem. However, our RSM is different than the one in classic simulation optimization literature, where one usually assumes known environments; for example, in an inventory optimization problem, demands and lead times follow some distributions with some estimated parameters.

This estimation is usually achieved through the analysis of historical data. Then, considering a specific environment (that is, distributions with specific parameters), one finds the optimal operating conditions for this inventory system, without exploring systematically other possible environments. Obviously, if the true environment happens to be different from the one considered in simulation optimization procedure, then the optimal operating conditions may become sub-optimal. In the chapters to follow, we explain our Taguchian RSM, which takes into account several possible environments in a systematic way.

2. TAGUCHIAN APPROACH TO RESPONSE SURFACE METHODOLOGY

2.1. INTRODUCTION

In classic simulation optimization, we usually minimize the expectation of a random response, say w, for which an explicit mathematical formulation is not available, and therefore the expectation is estimated through simulation [57].

This type of problems can be formulated as follows:

$$\text{minimize } \mathrm{E}\left[w \middle| d \right] \tag{2.1}$$

where d is the vector of control variables. An example of (2.1) is an inventory problem, where w is the sum of ordering, inventory-carrying, and penalty costs for back-ordered demands.

There are two disadvantages related to the formulation in (2.1). First, (2.1) is risk-neutral; that is, w is minimized on average without taking into account, for example, its estimated variance. To introduce the second disadvantage, we need to consider the simulation study that estimates w-for example, an inventory simulation where we assume demands follow an exponential distribution with mean $1/\lambda$. Now the second disadvantage is that the simulation study is done considering a single point estimate for this λ (known environment).

In Response Surface Methodology (RSM), the risk-neutrality problem of (2.1) was detected by Myers and Carter [58] who introduced the dual response surface (DRS) approach. This approach was further popularized by Vining and Myers [59], and since

then it has received a great deal of attention from researchers including Fan and Del Castillo [60], Yang, Kuo, and Chou [61], Lee and Park [62], Köksoy and Yalçinöz [63], and Dellino et al.[56].

In all papers mentioned in the previous paragraph except Dellino et al. [56], the DRS approach was applied to real-life problems; Dellino et al. [56] considered deterministic simulation. Therefore the main contribution of this thesis is to adapt Taguchian RSM to discrete-event simulation studies. Furthermore, we systematically explore the environmental variables by letting them to take their values from some intervals.

2.2. LITERATURE REVIEW

During the last couple of decades, robust design methodology has received increasing attention from engineers and researchers, due to the need of designing, formulating, developing, and analyzing new products or improving the existing ones.

In the early 1980's, Taguchi proposed the robust design approach. Since then, the Taguchian robust design methodology and its extensions have been widely used in many industrial applications to improve product quality and production methods. Applications of robust design to various engineering problems in the automotive industry, plastic technology, process industry, and information technology can be found in Bendell, Disney and Pridmore, and Dehnad [64, 65]. For robust process design, we refer to Taguchi and Wu; Taguchi; Box; Phadke; Welch, Yu, kang and Sacks; Shoemaker, Tsui and Wu; Pledger; Borkowski and Lucas; Wu and Hannada; and Myers and Montgomery [66, 67, 68, 69, 70, 71, 72, 73, 74, 55]. Finally, for the applications of Taguchi's approach to quality management and artificial neural networks, we refer to Lin, Sullivan and Taguchi; and Lin and Tseng, respectively [75, 76].

Taguchi defines robust design as a product whose performance is minimally sensitive to factors causing variability (at the lowest possible cost) [77] and to achieve desirable product quality by design, he suggests a three stage process [78]:

- System design, which is related to the conceptualization and synthesis of a product or process to be used;
- Parameter design, which is related to finding the appropriate design factor levels to make the system less sensitive to variations in uncontrollable noise factors (that is, to make the system robust);
- Tolerance design, which occurs when the tolerances for the products or process are established to minimize the sum of the manufacturing and lifetime costs of the product or process.

Two important tools used in the parameter design of Taguchi are orthogonal arrays and signal-to-noise ratios. Orthogonal arrays are used to test the different levels of each of the control factors, and signal-to-noise ratios as a quality indicator.

The main difference of Taguchi's method compared to ordinary optimization lies in the accounting for performance variations due to noise factors beyond the control of the designer [2]. That is, Taguchi's method selects the levels of the controllable factors to obtain the optimal operating conditions of control factors by reducing the variability around a nominal value of a quality characteristic of interest, and at the same time it keeps the process mean at the customer-identified target value.

Although Taguchi has had tremendous impacts on robust product and process designs, his approach has received much criticism, particularly because of the use of crossed orthogonal arrays as experimental designs and signal-to-noise-ratios. Several shortcomings of Taguchi's approach have been pointed out by Box; Vining and Myers; Pignatiello and Ramberg; Myers, Khuri and Vining; Myers and Montgomery; Leon, Shoemaker, and Kackar; Box, Bisgaard, and Fung; Nair et al. and Tsui [79, 80, 81, 82, 83, 84, 54, 85]. As a result, several researches have provided alternative methods to Taguchi's robust parameter design.

The concept of robustness was introduced by Myers and Carter [58] to RSM methodology through the DRS approach to model their problem. Their objective was to find the optimal operating settings that optimize a primary response, subject to the

condition that a secondary response takes on a desirable value. Since then, many researchers have focused on the dual response approach. These approaches can be classified as follows. Some researchers including Myers and Carter [58]; Fan and Del Castillo [60]; Tang and Xu [86]; Ross, Osborne, and George [87]; Köksoy and Doğanaksoy [88]; Yang, Kuo, and Chou [89]; Peterson and Kuhn [90]; Jeong, Kim, and Chang [91]; Yeniay, Unal, and Lepsch [92]; Lee and Park [93]; Lee, Park, and Cho [94], and Köksoy and Yalçınöz [95] considered only control factors when they approximated their unknown responses through first or second-order regression polynomials. On the other hand, Miró-Quesada and Del Castillo [96]; Miró-Quesada and Del Castillo [97]; Myers, Brenneman, and Myers [98]; Rajagopal, Del Castillo, and Peterson [99], Giovagnoli and Romano [100]; and Dellino et al. [56] considered both noise and control factors. Below, we summarize the contributions of these papers to the robust response surface methodology.

The abundant literature on RSM about how to seek optimal operating settings for dual response systems using various optimization approaches neglects the inherent sampling variability of the fitted responses. Therefore, Fan and Del Castillo [60] introduced Monte Carlo sampling to the dual response approach and constructed an optimal region in the control factor space, which provides more useful information to a process engineer than a single expected optimal solution.

Tang and Xu proposed a goal programming approach to optimize a dual response system. Their formulation is general enough to include some of the existing methods as special case [86].

Ross et al. presented a mathematically rigorous approach for incorporating decision-maker preferences. By interpreting the Lagrangian as a value function and the Lagrange multiplier as a preference ratio, they explored candidate solutions that reflect decision-maker preferences [87].

Taguchi's robust parameter design calls for simultaneous optimization of the mean and standard deviation responses. The dual response optimization procedures have been

adapted to achieve this goal by taking into account both the mean and standard deviation responses. The popular formulations of the dual response problem typically impose a restriction on the value of the secondary response (i.e., keeping the standard deviation below a specified value) and optimize the primary response (i.e., maximize or minimize the mean). Restrictions on the secondary response, however, may rule out better conditions, since an acceptable value for the secondary response is usually unknown. In fact, process conditions that result in a smaller standard deviation are often preferable. A more flexible formulation of the problem can be achieved by considering the secondary response as another primary response. Therefore, Köksoy and Doğanaksoy introduced Pareto optimal solutions, which give more flexibility to the decision-makers in exploring alternative solutions [88]. Furthermore, Köksoy and Yalçınöz again followed Pareto optimal solutions strategy, but this time they solved the DRS problem through a genetic algorithm [95].

Yang et al. solved a multiresponse simulation problem by using a dual response system and scatter search method. Their proposed dual response system constructs a response surface for each response [89]. It then transforms the dual response system into a standard nonlinear programming formulation. The transformation treats the secondary response as a constraint. In addition, the sample variance from simulation replications is considered simultaneously by adding search area constraints to variance. Their proposed scatter search method uses scatter search algorithms as an embedded mechanism in a simulation program to guide the solution search process.

Peterson and Kuhn proposed an approach to doing a ridge analysis for optimizing a response surface in the presence of noise variables [90]. Their approach allows an investigator to explore factor combinations that lower the mean squared error about a target value, while at the same time keeping track of how much the mean response differs from the target value. Their approach also allows an investigator to compute a simultaneous confidence band about the root mean squared error about a target value.

The dual response surface optimization simultaneously considers the mean and the standard deviation of a response. The minimization of the mean squared error is a

simple, yet effective, approach in DRS optimization. The bias and variance components of the mean squared error need to be weighted properly if they are not in the same importance in the given problem situation. To date, the relative weights of bias and variance have been equally set or determined only by the data. However, the weights should be determined in accordance with the tradeoffs on various factors in quality and costs. Therefore, Jeong et al. (2005) proposed a systematic method to determine the weights of bias and variance in accordance with a decision-maker's preference structure regarding the tradeoffs [91].

Yeniay et al. utilized the DRS approach to quantify variability in critical performance characteristics during conceptual design phase of a launch vehicle [92]. Using design of experiments methods and disciplinary design analysis codes, dual response surfaces are constructed for the mean and standard deviation to quantify variability in vehicle weight and sizing analysis. Next, an optimum solution is sought to minimize variability subject to a constraint on mean weight.

In robust design, a commonly used assumption behind the data collection procedure is that all the data are fully observed. However, in many industrial experiments, interval censored observations are frequently available in addition to the fully observed observations.

Therefore, Lee and Park calculated the optimal operating conditions for the process based on a dual response approach using incomplete data [93]. In their novel approach, they estimate the process mean and variance with incomplete data. Thus, it is possible to find the optimal operating conditions using all of the information available.

Robust design uses the ordinary least squares method to obtain adequate response functions for the process mean and variance by assuming that experimental data are normally distributed and that there is no major contamination in the data set. Under these assumptions, the sample mean and variance are often used to estimate the process mean and variance. In practice, the above assumptions are not always satisfied. When these assumptions are violated, one can alternatively use the sample median and median

absolute deviation to estimate the process mean and variance. However, the median and median absolute deviations both suffer from a lack of efficiency under the normal distribution, although they are fairly outlier-resistant. To remedy this problem, Lee et al. proposed new robust design methods based on a highly efficient and outlier resistant estimator [94].

Miró-Quesada and Del Castillo proposed an extension to the dual response approach to robust parameter design for the case of multiple responses [96]. Their methodology provides unbiased estimates of the process covariance matrix and of the vector of expected values using parameter estimates from a multivariate regression fit.

Miró-Quesada and Del Castillo studied the prediction properties of models used in the dual response approach to robust parameter design, and they proposed two procedures that improve the performance of the approach [97]. Their first procedure suggests scaling of the noise variables to reduce the expected mean squared error of the variance model, based on the concept that the range of the noise variables used in the experimental design should contain most of their distribution. However, it is shown that such scaling does not alter the variance contribution of the noise factors, which is fundamental for robust parameter design. Their second procedure combines the variance due to the noise factors with the variance due to the prediction error of the fitted model, thus considering all sources of variability present in the problem. An unbiased estimator of this combined variance is developed.

Robust parameter design has been studied and applied, in most cases, assuming a linear model under standard assumptions. More recently, robust parameter design has been considered in a generalized linear model setting. Myers et al. applied a general dual response approach when using robust parameter design in the case of a generalized linear model [98]. They motivated the need for exploring both the process mean and process variance by discussing situations when a compromise between the two is necessary.

The uncertainty of the model form is typically neglected in process optimization studies. In addition, not taking into account the existence of noise factors and nonnormal errors may invalidate the conclusions of such studies. Rajagopal et al. presented a Bayesian approach to model robust process optimization in the presence of noise factors and nonnormal error terms [99]. Their paper extended the idea of model form-robustness using a Bayesian predictive approach to cases where there is uncertainty due to the distributional assumptions of the errors.

The existing procedures for robust design, devised for physical experiments, may be too limiting when the system can be simulated by a computer model. Therefore, Giovagnoli and Romano introduced a modification of the DRS modeling, which incorporates the option of stochastically simulating some of the noise factors when their probabilistic behavior is known [100]. Their method generalizes both the crossed and the combined array approaches and finds a natural application to integrated parameter and tolerance design.

Optimization of simulated systems is tackled by many methods, but most methods assume known environments. Therefore, Dellino et al. [56] developed a robust methodology for uncertain environments. Their methodology uses Taguchi's view of the uncertain world, but replaces his statistical techniques by Response Surface Methodology.

2.3. TAGUCHI'S ROBUST DESIGN

Robust means that the process or product performs consistently and is relatively insensitive to the factors that are difficult to control. Hence, Robust Design approach aims to provide a method for designing products and processes that are minimally impacted by external forces, such as environment, client use or manufacturing-based factors named as uncontrollable factors .

As mentionned in the literature review part, Taguchi suggests a three stage process (*system design, parameter design, tolerance design*) in order to minimize the

process/product variation and to design robust and flexible processes/products that are adaptable to environmental conditions.

According to Taguchi, there are two types of factors that affect a product's functional characteristic which can be found in Figure 2.1: *control factors* and *noise factors* [56].

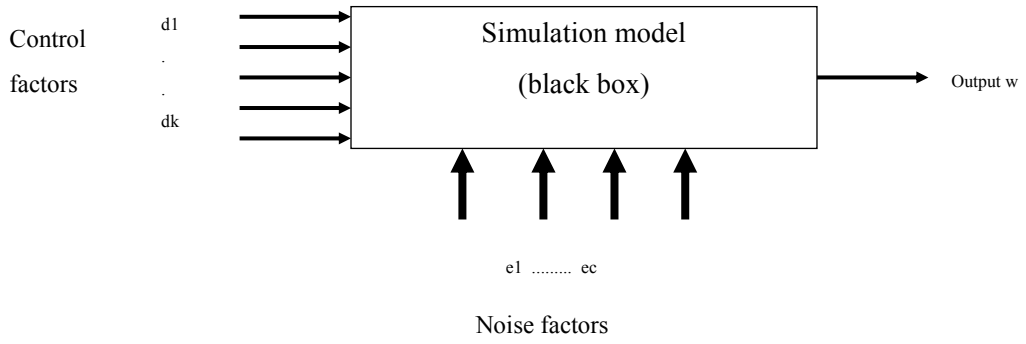

Figure 2.1: Taguchi factors

The first type of factors are under the control of the users, and the second type of factors are difficult or impossible or too expensive to control. Hence, parameter design seeks to identify settings of the control factors which make the product insensitive to variations in the noise factors, i.e., make the product more robust, without actually eliminating the causes of variation.

Design of experiments techniques, specifically Orthogonal Arrays (OAs), are employed in Taguchi's approach to systematically vary and test the different levels of each of the control factors. A complete listing of OAs can be found in text such as Phadke [69].

To implement robust design, Taguchi advocates the use of an "inner array" and "outer array" approach. The "inner array" consists of the OA that contains the control factor settings; the "outer array" consists of the OA that contains the noise factors and their settings which are under investigation. The combination of the "inner array" and "outer array" constitutes what is called the "product array" or "complete parameter design layout". The product array is used to systematically test various combinations of the control factor settings over all combinations of noise factors after which the mean

response and standard deviation may be approximated for each run using the following equations.

- Mean response: $$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.2}$$

- Standard deviation: $$S = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \bar{y})^2}{n-1}} \tag{2.3}$$

The preferred parameter settings are then determined through analysis of the "signal-to-noise" (SN) ratio where factor levels that maximize the appropriate SN ratio are optimal. There are three standard types of SN ratios depending on the desired performance response [78]:

- Smaller the better (for making the system response as small as possible):

$$SN_S = -10\log\left(\frac{1}{n}\sum_{i=1}^{n} y_i^2\right) \tag{2.4}$$

- Nominal the best (for reducing variability around a target):

$$SN_T = 10\log\left(\frac{\bar{y}^2}{S^2}\right) \tag{2.5}$$

- Larger the better (for making the system response as large as possible):

$$SN_L = -10\log\left(\frac{1}{n}\sum_{i=1}^{n} \frac{1}{y_i^2}\right) \tag{2.6}$$

These SN ratios are derived from the quadratic loss function and are expressed in a decibel scale.

Once all of the SN ratios have been computed for each run of an experiment, Taguchi advocates a graphical approach to analyze the data. In the graphical approach, the SN ratios and average responses are plotted for each factor against each of its levels. The graphs are then examined to "pick the winner," i.e., pick the factor level which (1) best maximize SN and (2) bring the mean on target (or maximize or minimize the mean, as the case may be).

Finally, confirmation tests should be run at the "optimal" product settings to verify that the predicted performance is actually realized. A demonstration of Taguchi's approach to parameter design can be found in Figure 2.2.

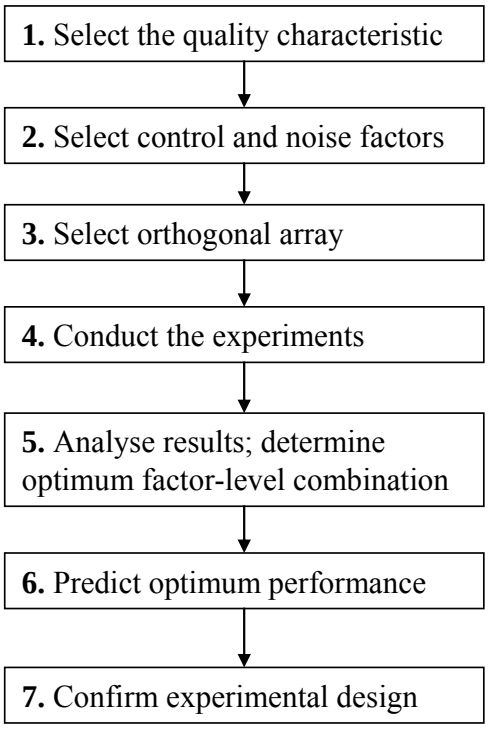

Figure 2.2: Main Steps of Taguchi Robust Design

Taguchi method is a simple and usefull tool, but there are also some drawbacks which require to be compansated with other optimization methodologies. These drawbacks are:

- When the number of design parameters increases its computational requirements becomes unaffordable.
- Taguchi method has no capability to handle multiple performance measures simultaneously.

2.4. TAGUCHIAN RESPONSE SURFACE METHODOLOGY

In this section, we introduce the steps of Taguchian RSM. We assume that the initial point is in a neighborhood of an optimal solution of the problem in (2.1), say d^*, which was already found by a classic simulation optimization algorithm. Hence, starting from the neighborhood of d^*, our Taguchian RSM will search for a robust optimal solution in one shot, rather than searching iteratively over the global feasible area.

Taguchian RSM consists of four steps, namely selecting a design type of experiments, fitting a regression metamodel to the realizations of the random response w in (2.1) and estimating its variance, checking the validity of the fitted metamodel, and minimizing a risk-averse transformation of the problem in (2.1). We will describe these steps in detail later in this section. Our description is in general in line with the one in Dellino et al. [56]; whenever there is a difference, we will make it clear in the text.

In Step 2 of Taguchian RSM, Myers and Montgomery [55] suggested to approximate w in (2.1) through

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j d_j + \sum_{j=1}^{k} \sum_{j' \geq j}^{k} \beta_{j;j'} d_j d_{j'} + \sum_{g=1}^{c} \gamma_g e_g + \sum_{j=1}^{k} \sum_{g=1}^{c} \delta_{j;g} d_j e_g + \varepsilon \qquad (2.7)$$

$$= \beta_0 + \beta^T d + d^T B d + \gamma^T e + d^T \Delta e + \varepsilon \qquad (2.8)$$

where y is the regression predictor, β_0 is the overall mean, the $k\beta_j$'s are the main effects of the controllable variables, the $\frac{k^2}{2}\beta_{j;j'}$'s are the two-factor interactions ($j \neq j'$) and the purely quadratic effects ($j = j'$), the $c\gamma_g$'s are the main effects of the environmental variables, and the $kc\delta_{j;g}$'s are the control-by-noise interactions. This makes a total of $q = 1 + k + \frac{k^2}{2} + c + kc$ unknown coefficients to be estimated in (2.7). Furthermore, the ε is the residual which is supposed to satisfy the white noise assumption; that is, ε is normally, independently, and identically distributed with mean zero ($\mu_\varepsilon = 0$) and constant variance σ_ε^2 [55]. $\mu_\varepsilon = 0$ implies that the metamodel in (2.5) has no lack-of-fit, which will be investigated in Step 3. Finally, (2.8) is obtained by simply rewriting (2.7) in matrix notation.

Myers and Montgomery assumed the following for the vector of the environmental variables e in (2.8): $E(e) = 0$ and $\text{cov}(e) = \sigma_e^2 \text{I}$, where $\text{cov}(e)$ and I denote the covariance matrix of e and the identity matrix, respectively [55]. We, however, prefer to replace their assumption with the more realistic one in Dellino et al.: $E(e) = \mu_e$ and $\text{cov}(e) = \Omega_e$, where both μ_e and Ω_e are assumed to be known [56]. Under Dellino et al.'s assumption, the mean and variance of y are given by

$$E(y) = \beta_0 + \beta^T d + d^T B d + \gamma^T \mu_e + d^T \Delta \mu_e \tag{2.9}$$

$$\text{var}(y) = \text{var}\left[\left(\gamma^T + d^T\Delta\right)e\right] + \sigma_\varepsilon^2 = \left(\gamma^T + d^T\Delta\right)\Omega_e\left(\gamma + \Delta^T d\right) + \sigma_\varepsilon^2 \tag{2.10}$$

provided that e and ε are independent [56]. Estimates of (2.7) and (2.10) will be used when we minimize a risk-averse version of (2.1) in Step 4.

Now, we can detail each step of Taguchian RSM, as follows. These steps will be used only once when we apply Taguchian RSM to an inventory example in the next section (i.e., one shot approach).

Step 1, Select a design type of experiments: To fit the metamodel in (2.7), simulation practitioners usually prefer a central composite design (CCD), which is defined as follows. One part of a CCD consists of a two-level factorial design that may be fractional-provided this fractional has a resolution at least V - since a resolution V (R_V) design gives unbiased ordinary least squares (OLS) estimators of all main effects and all two-factor interactions-provided all other effects are negligible. Furthermore, to estimate all purely quadratic effects, a CCD augments a R_V design by (i) the central design point and (ii) $2(k+c)$ axial design points [101]. In our CCD, we have only $2k$ axial design points in addition to the central design point, since there are no purely quadratic effects of the environmental variables in (2.7).

Step 2, Fit the metamodel and estimate variance: Let n be the total number of input combinations, which depends on the selected design type in Step 1. We simulate m_l replicates at the lth design point $(d_l^T, e_l^T)^T$ $l = (1,...,n)$, which give m_l identically and independently distributed simulated responses at that point. Furthermore, we do not use common seeds across the n input combinations, to make the resulting total number of runs $N = m_1 + ... + m_l + ... + m_n$ independent. Our main reason for avoiding common seeds is the well-known synchronization problem of discrete-event simulation studies [102]. By this way, we obtain the $N \times 1$ vector $\hat{w}$ of simulated responses.

Let ζ be the $q \times 1$ vector whose components are the unknown coefficients in (2.7). If the ε satisfies the white noise assumption, then the best linear unbiased estimator of ζ is given by its OLS estimator:

$$\hat{\zeta} = \left(X^T X \right)^{-1} X^T \hat{w} \tag{2.11}$$

where X is the $N \times q$ matrix of explanatory variables. In (2.11), we use the original values of d and e; because the inversion in (2.11) may cause numerical instabilities, Dellino et al. used their standardized (or coded) values [56].

Assuming that the variance of w is constant across the n input combinations, the covariance matrix of $\hat{\zeta}$ is given by

$$\hat{\Psi} = \hat{\sigma}_{\hat{w}}^2 \left(X^T X \right)^{-1} \tag{2.12}$$

where $\sigma_{\hat{w}}^2$ is estimated through mean squared residuals [101]:

$$\hat{\sigma}_{\hat{w}}^2 = \frac{\left(\hat{w} - \hat{y} \right)^T \left(\hat{w} - \hat{y} \right)}{N - q} \tag{2.13}$$

provided $\hat{y} = X\hat{\zeta}$ and $N > q$. Considering the dependence of (2.10) on d, the constant variance assumption may not be very realistic; we left this issue as a future work.

For (2.11) to be multivariate normally distributed, Dellino et al., who considered deterministic simulation, assumed that e is multivariate normally distributed [56]. We, however, consider stochastic simulation where we can make a large number of runs; that is, $N \to \infty$ such that $m_l \to \infty$ for each input combination l. Then, under some conditions, $\left\{ \sqrt{N} \left(\hat{\zeta} - \zeta \right) \right\}$ has asymptotically a multivariate normal distribution with zero mean vector and covariance matrix Ψ with Ψ given by (2.12) replacing $\hat{\sigma}_{\hat{w}}^2$ with $\sigma_{\hat{w}}^2$ [103]. In practice, ζ and $\sigma_{\hat{w}}^2$ are computed through (2.11) and (2.13) for large N. Therefore, we do not need to assume multivariate normality for e. Furthermore, the asymptotic multivariate normality of $\hat{\zeta}$ enables us to apply classic F-test for lack-of-fit in the following step.

Step 3, Test the validity of the fitted metamodel: We test the following null hypothesis:

$$H_0 : E(w) = E(y). \tag{2.14}$$

If H_0 is rejected, the approach in classic RSM is to switch alternative regression metamodels using some transformations of d; in this thesis, we do not discuss solutions when there is lack-of-fit, but we refer to Irrizary, Kuhl, Lada, Subramanian and Wilson for such solutions [104]. In case H_0 is not rejected, Step 4 will be performed.

To test H_0, we introduce the following classic F-statistic:

$$F_{\nu_1,\nu_2} = \frac{SS_{LOF}/\nu_1}{SS_{PE}/\nu_2} \tag{2.15}$$

where SS_{LOF} is the lack-of-fit sum of squares, SS_{PE} is the sum of squared pure errors, and $\nu_1 = n - q$ and $\nu_2 = N - n$ are the degrees of freedom of SS_{LOF} and SS_{PE}, respectively. H_0 is rejected if F_{ν_1,ν_2} exceeds a prespecified critical value F_{α,ν_1,ν_2}, where α is the type-I error rate.

To compute F_{ν_1,ν_2}, we introduce the sum of squared residuals, SS_R, since SS_{LOF} is given by $SS_{LOF} = SS_R - SS_{PE}$. These SS_R and SS_{PE} can be computed through

$$SS_R = \left(\hat{w} - \hat{y}\right)^T \left(\hat{w} - \hat{y}\right) \tag{2.16}$$

and

$$SS_{PE} = \left(\hat{w} - \bar{\hat{w}}\right)^T \left(\hat{w} - \bar{\hat{w}}\right) \tag{2.17}$$

where $\bar{\hat{w}}$ is the $N \times 1$ vector of responses averaged over the m_l replicates; i.e., its first m_1 rows consist of $\bar{\hat{w}}_{m_1}$, which denotes the average of the first m_1 components of $\hat{w}$, its next m_2 rows consist of $\bar{\hat{w}}_{m_2}$, which denotes the average of the next m_2 components of $\hat{w}$, etc.

Finally, to test H_0, Dellino et al. used leave-one-out cross-validation. Our F-test approach in (2.15) is more standard in regression analysis [56].

Step 4, Minimize the risk-averse problem: To obtain a risk-averse formulation of the problem in (2.1), we first estimate (2.9) and (2.10) by

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}^T d + d^T \hat{B} d + \hat{\gamma}^T \mu_e + d^T \hat{\Delta} \mu_e \tag{2.18}$$

$$\widehat{\text{var}}(y) = \left(\hat{\gamma}^T + d^T \hat{\Delta}\right) \Omega_e \left(\hat{\gamma} + \hat{\Delta}^T d\right) + \hat{\sigma}_\varepsilon^2 \tag{2.19}$$

Note that the variance estimator in (2.19) is biased; Myers and Montgomery gave an unbiased estimator for $\text{var}(y)$ [55], but for simplicity we use (2.19). Furthermore, we estimate $\hat{\sigma}_\varepsilon^2$ through (2.13).

Now a risk-averse formulation is given by *the smaller the better* approach:

$$\begin{aligned} &\text{minimize } \hat{y} \\ &\text{subject to } \widehat{\text{var}}(y) \le \tau \end{aligned} \tag{2.20}$$

where τ denotes a threshold value. We will change this τ over a finite interval and solve (2.20) each time with the new τ to observe the price of taking risk.

2.5. (s, S) INVENTORY EXAMPLE

In this section, we applied Taguchian RSM to an (s, S) inventory example investigated by Bashyam and Fu [105]. The computer program was coded in Matlab 7.6, the optimization in (2.20) was performed through the built-in function *fmincon* in Matlab, and could be analyzed in Appendix A and B.

2.5.1. Problem Definition

Bashyam and Fu [105] considered an infinite horizon periodic review inventory system with continuous-valued independently and identically distributed demands and full backlogging.

Orders are received at the beginning of the period; the demand for the period is subtracted out, then an order review is carried out at the end of the period. The inventory level in period n (W_n) is defined as the on hand stock minus backorders, and observed after demand subtraction, and the inventory position (I_n) is the inventory level plus any outstanding orders [105].

Ordering decisions are made according to the wellknown (s, S) policy:

If $I_n < s$: an order for the amount $S - I_n$ is placed

O/w, no action is taken

The lead times L_i for orders placed are assumed to be integer valued i.i.d. random variables. Under their convention, an order with lead time l placed in period n will arrive at the beginning of period $n + l + 1$.

The performance of the system is evaluated by a cost function and a service level measure, where the cost measure considers only setup and holding costs, and the service level measure tracks the extent of backlogging in the system.

2.5.2. Application

Like Bashyam and Fu, we assumed an infinite horizon, periodic review inventory system with exponentially distributed demands with mean 100 and Poisson distributed order lead times with mean 6, and full backlogging of orders [106]. The basic sequence of events in each period is as follows: orders are received at the beginning of the period, the demand for the period is subtracted, and order review is done at the end of the

period. An order is placed when the inventory position falls below the reorder level *s*; the order amount is the difference between the order up to level *S* and the current inventory position.

Furthermore, Bashyam and Fu set per order setup cost to 36, per unit order cost to 2, and per period per unit holding cost to 1; we used the same cost values in our simulation experiments [105]. Moreover, Bashyam and Fu had a service level constraint, which we did omit, since this would require considering multiple responses; we left this issue as a future work. We, however, had the following deterministic constraint in addition to (2.20): $s \le S$.

Angün (2008) found an estimated optimal solution of Bashyam and Fu's problem as $\left(s^*, S^*\right)^T = \left(1160, 1212\right)^T$ with an estimated cost of 647.15 [106]. Starting from this estimated optimal solution, our goal is to find a robust optimal solution. The random response w to be minimized is the total costs, namely the sum of order setup, ordering, and holding costs. The environmental variable e is the mean demand.

In our experiment, the factorial part of CCD was given by a 23 design with $980 \le s \le 1340$, $1019 \le S \le 1405$ and $80 \le e \le 120$, the central point by s = 1160, S = 1212, and e = 100, the two positive axial points by s = 1424, S = 1212, and e = 100, and s = 1160, S = 1476, and e = 100, and the two negative axial points by s = 896, S = 1212, and e = 100, and s = 1160, S = 948, and e = 100, all expressed in the original variables; obviously, n = 13. Notice that the low (80) and high (120) values for e were chosen as $100 \pm \sqrt{\Omega_e}/5$, where $\Omega_e = 100^2$ is the variance of e. Furthermore, the number of replicates at each input combination l was chosen as $m_l = 30$ so that $N = 390$. Each simulation run was simulated for 2500 periods, and the type-I error rate α for lack-of-fit test was chosen as $\alpha = 1\%$. The computer program was coded in Matlab 7.6, and could be analyzed in Appendix C.

Table 2.1.: Optimal regression predictors $\hat{y}^*$ and their risks $\sqrt{\widehat{\text{var}}(\hat{y}^*)}$ obtained through solving (2.20) with $s \leq S$

$\hat{y}^*$	$\sqrt{\widehat{\text{var}}(\hat{y}^*)}$	$\hat{y}^*$	$\sqrt{\widehat{\text{var}}(\hat{y}^*)}$	$\hat{y}^*$	$\sqrt{\widehat{\text{var}}(\hat{y}^*)}$
-408	232	-409	233	-410	234
-411	236	-413	237	-415	238
-416	240	-418	241	-419	242

We solved (2.20) subject to $s \leq S$ by changing the target value τ over the interval [50,000; 58,000]; that is, we started with τ = 50,000, solved the problem, increased τ by 1000, and resolved the new problem up to and including τ = 58,000. Our numerical results are presented in Table 2.1; we rounded up all decimals to the nearest integers.

According to the results in Table 2.1, the lower the cost is, the higher the risk becomes, which is in accordance with the common sense. In particular, the lowest cost, namely 419, has the highest risk, namely 242. A further remark that should be made is that all optimal costs in Table 2.1 are lower than the one in Angün (2008) as seen in Table 2.2. simply because in this thesis, we did not consider the service level constraint [106].

Table 2.2.: Numerical results for the 0.50 quantile ('mean') of 100 estimated solutions for the inventory problem

iteration	iterate $(\hat{s}, \hat{S})$	search direction $\hat{d}$	step size λ	$H_0^{(1)}$	$H_0^{(2)}$
0	(2100, 2300)	(-0.7045, -0.7097)	1703.4		
1	(900, 1091)			reject	fail to reject
2	(1500, 1695.5)			reject	reject
3	(1200, 1393.3)	(-0.5973, -0.8020)	171.3	reject	reject
4	(1097.6, 1255.9)			reject	fail to reject
5	(1148.8, 1324.6)	(-0.4816, -0.8764)	100.9	reject	reject
6	(1123.2, 1290.2)			reject	fail to reject
7	(1100.2, 1236.1)			reject	fail to reject
8	(1124.5, 1280.3)			reject	fail to reject
9	(1136.7, 1302.4)			reject	reject

3. CALL CENTER APPLICATION

In this chapter, we will apply our Taguchian RSM approach to a more complex example which is a call center problem modeled in Arena. The results taken from the execution of the model will be used in our optimization algorithm.

3.1. PROBLEM DEFINITION

The generic call center system described in detail in the Simulation with Arena book provides a central number in an organization that customers call for technical support, sales information, and order status. This central number feeds 26 trunk lines. If all 26 lines are in use, a caller gets a busy signal; or an answered caller hears a recording describing three options: transfer to technical support, sales information, or order-status inquiry [107].

Below a brief description of each option is given and further details regarding waiting times, product types' request statistics, call duration estimates etc. can be found in Appendix D.

Technical Support Calls

If the caller chooses technical support, he/she gets a second recording asking which of three product types he/she is using:

- Product type 1? (25% of technical support callers)
- Product type 2? (34% of technical support callers)
- Product type 3? (41% of technical support callers)

If a qualified technical support person is available for chosen product type, the call is automatically routed to that person for immediate service. If not, the call is placed in an electronic queue until a support person is available. Upon completion of the call, the customer exits the system. However, four percent of these technical support calls needs further assistance after completion of the call. The questions raised by these callers are forwarded to another technical group, outside the boundaries of the defined model that prepares a response. The resulting response is sent back to the same technical support person who took the original call. This person then calls the customer back. These calls require the use of one of the 26 trunk lines and takes priority over incoming calls. If a returned call is not completed on the same day the original call was received, it's carried over to the next day. A demonstration of the Generic Call Center System can be found in Figure 3.1.

Generic Call Center System

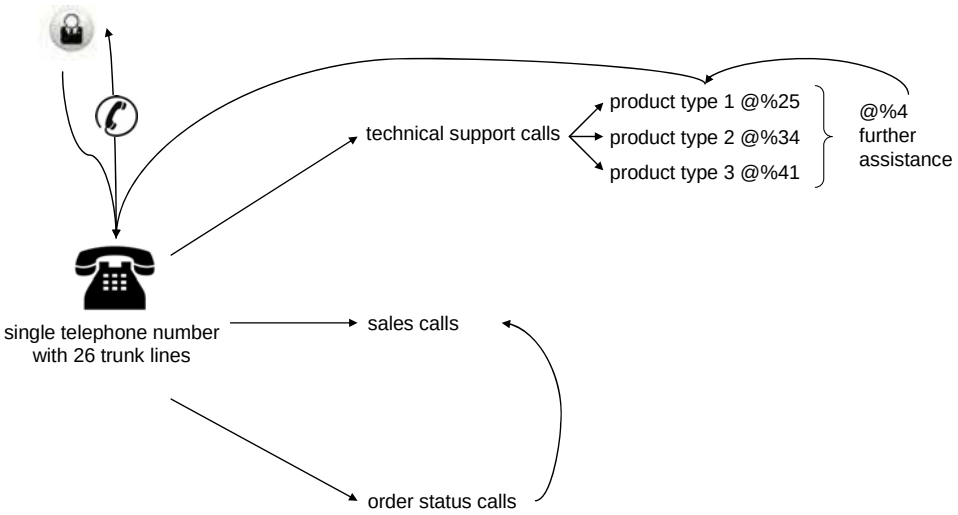

Figure 3.1: Generic Call Center System

There are 11 technical support people variously qualified for the three different product lines. Some people only qualified in one line, and some on two or maybe all three lines. Detailed stuffing description and schedule information can be found in Appendix D.

Sales Calls

These calls are automatically routed to the sales staff which is separated from technical support staff. There are seven sales people with the staggered daily schedules defined in number of people @ time period in minutes. If a salesperson is not available, the caller is waited on the line. Upon completion of the call, the customer exist the system.

Order-Status Calls

These calls are automatically handled by the phone system, and there is no limit on the number handled at a time (but still limited by the 26 trunk lines). After the call some of these callers take the option to talk to a real person and the rest exits the system. And these calls:

- are routed to the sales staff
- have the same priority as incoming sales calls.

and then customers exit the system.

The call center operates from 8 a.m. to 6 p.m., and a small proportion of the staff stays until 7 p.m. Incoming calls shut out after 6 p.m., but all calls that entered before 6 p.m. are answered. The call arrival rate varies substantially over the day, and is expressed in calls per hour for each 30-minute period during which the system is open. All technical support employees work an eight-hour day with 30 minutes off for lunch (lunch is not included in the eight hours). Top-level Model View of the Call Center Model can be found in Figure 3.2.

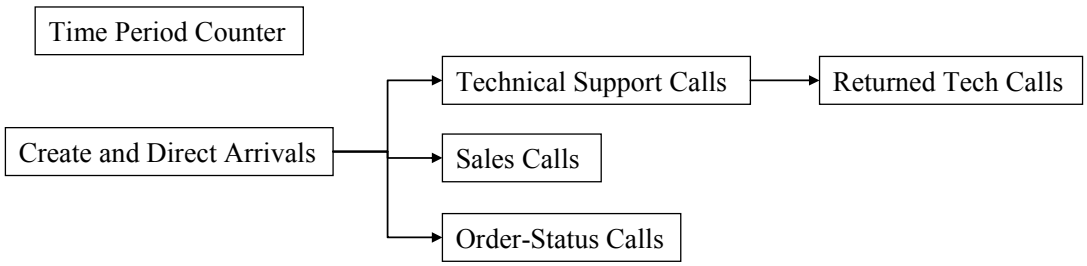

Figure 3.2: Top-level Model View of the Call Center Model

3.2. APPLICATION #1

Based on the defined model, our objective is to find an optimal solution $(x_1^*, x_2^*, x_3^*, e_1^*)$, where x_1 is the number of trunk lines, x_2 is the additional sales staff, x_3 is all product support staff (support people qualified for all product types) and e_1 is the recording delay of technical support calls having the uniform distribution $U(0.1, 0.5)$ as an environmental factor, all of which minimize the total system cost. These decision variables have the following constraints: $26 \le x_1 \le 50$, $x_2 + x_3 \le 15$, $0.5 - r \le e_1 \le 0.5 + r$ where r is the radius. The objective function is the expected total system cost of new technical people and new sales staff, and new trunk lines.

In the first step of our approach, we select the design type of experiments. As in our previous example, we prefer to use a CCD design and since we have 4 design factors, we select a 2^4 CCD design with $2k = 6$ axial design points and a central design point which are given in detail in Appendix D. Therefore, we have $n = 23$ input combinations.

To fit the metamodel and estimate its variance, we simulate $m_l = 10$ replicates at each design point and obtain 10 identically and independently distributed simulated responses at each point. By this way, we obtain the 230×1 vector $\hat{w}$ of simulated responses.

Type-I error rate α for lack-of-fit test is chosen as $\alpha = 1\%$. The computer program was coded in Matlab 7.6, and could be analyzed in Appendix E.

Finally, in order to minimize the risk averse problem, we first estimate (3.1):

$$\begin{aligned}\hat{y} = {} & \widehat{\beta_0} + \widehat{\beta_1} x_1 + \widehat{\beta_2} x_2 + \widehat{\beta_3} x_3 + \hat{\gamma}_1 e_1 + \widehat{\beta_{1,2}} x_1 x_2 + \widehat{\beta_{1,3}} x_1 x_3 + \widehat{\beta_{2,3}} x_2 x_3 \\ & + \widehat{\delta_{1,1}} e_1 x_1 + \widehat{\delta_{1,2}} e_1 x_2 + \widehat{\delta_{1,3}} e_1 x_3 + \widehat{\beta_{1,1}} x_1^2 + \widehat{\beta_{2,2}} x_2^2 + \widehat{\beta_{3,3}} x_3^2\end{aligned} \tag{3.1}$$

For this, we use our computer program which was coded in Matlab 7.6, and could be analyzed in Appendix F.

3.3. RESULTS #1

Based on the results taken from the execution of the Matlab codes, we observe $\hat{\beta}$ as below:

$$\hat{\beta} = \begin{bmatrix} -894160 \\ 46944.3 \\ 363.787 \\ 71341.3 \\ 431242 \\ -3.60577 \\ -1687.57 \\ 18.75 \\ -6989.71 \\ -187.5 \\ -50035.5 \\ -672.168 \\ -24.7372 \\ -229.493 \end{bmatrix}$$

The optimization of the problem is performed by using the *fmincon* function. Our minimization resulted with $exitflag = 1$ which means that the problem is solved at optimum. The optimum value for x is as below:

$$x = \begin{bmatrix} 26.000 \\ 0 \\ 0 \\ 0.4950 \end{bmatrix}$$

The objective value is equal to -4.4865e+003.

3.4. APPLICATION #2

Since the student version of Arena doesn't allow us to increase the number of decision variables, we therefore decided to add one more environmental factor e_2 in order to expand the original example.

Then, our objective becomes to find an optimal solution $(x_1^*, x_2^*, x_3^*, e_1^*, e_2^*)$, where x_1 is the number of trunk lines, x_2 is the additional sales staff, x_3 is all product support staff (support people qualified for all product types), e_1 is the recording delay of technical support calls having the uniform distribution $U(0.1, 0.5)$ and e_2 is the delay of create & direct arrivals part of the system having the uniform distribution U(0.1,0.6) as an environmental factor, all of which minimize the total system cost.

These decision variables have the same constraints as application #1. The objective function is the expected total system cost of new technical people and new sales staff, and new trunk lines.

In the first step of our approach, we select the design type of experiments. As in our previous example, we prefer to use a CCD design and since we have 5 design factors now, we select a 2^5 CCD design with $2k = 6$ axial design points and a central design point which are given in detail in Appendix G. Therefore, we have $n = 39$ input combinations.

To fit the metamodel and estimate its variance, we simulate $m_l = 10$ replicates at each design point and obtain 10 identically and independently distributed simulated responses at each point. By this way, we obtain the 390×1 vector $\hat{w}$ of simulated responses.

Type-I error rate α for lack-of-fit test is chosen as $\alpha = 1\%$. The computer program was coded in Matlab 7.6, and could be analyzed in Appendix H.

Finally, in order to minimize the risk averse problem, we first estimate (3.2):

$$\begin{aligned}\hat{y} &= \widehat{\beta_0} + \widehat{\beta_1}x_1 + \widehat{\beta_2}x_2 + \widehat{\beta_3}x_3 + \widehat{\beta_{1,2}}x_1x_2 + \widehat{\beta_{1,3}}x_1x_3 + \widehat{\beta_{2,3}}x_2x_3 \\ &+\hat{\gamma}_1e_1 + \widehat{\delta_{1,1}}e_1x_1 + \widehat{\delta_{1,2}}e_1x_2 + \widehat{\delta_{1,3}}e_1x_3 + \hat{\gamma}_2e_2 + \widehat{\delta_{2,1}}e_2x_1 + \widehat{\delta_{2,2}}e_2x_2 \\ &+\widehat{\delta_{2,3}}e_2x_3 + \widehat{\beta_{1,1}}x_1^2 + \widehat{\beta_{2,2}}x_2^2 + \widehat{\beta_{3,3}}x_3^2\end{aligned} \tag{3.2}$$

For this, we use our computer program which was coded in Matlab 7.6, and could be analyzed in Appendix I.

3.5. RESULTS #2

Based on the results taken from the execution of Matlab codes, we observe $\widehat{\beta}$ as below:

$$\widehat{\beta} = \begin{bmatrix} -2.00829e+006 \\ 43093.4 \\ 313036 \\ 41741.4 \\ 1.48e+006 \\ 1.18377e+006 \\ 6009.64 \\ -5114.92 \\ -31250.1 \\ -52091.2 \\ -312501 \\ 222195 \\ -40164.2 \\ -260418 \\ 178239 \\ 169.08 \\ -6.47815 \\ -200.209 \end{bmatrix}$$

The optimization of the problem is performed by using the *fmincon* function. Our minimization resulted with $exitflag = 4$ which means that the magnitude of the search direction is very small. The optimum value for x is as below:

$$x = \begin{bmatrix} 50.000 \\ 12.4348 \\ 2.5652 \\ 0.5050 \\ 0.6060 \end{bmatrix}$$

The objective value is equal to -5.2503e+006.

We also experiment our minimization algorithm with different starting points, and obtain same results; therefore, we can say that we reached a local optimum.

4. CONCLUSION

In this thesis, we presented the steps of Taguchian RSM, which consisted of selecting an experimental design type, approximating the random response by a regression metamodel and estimating its variance, performing a lack-of-fit test to check the validity of the metamodel, and minimizing a risk-averse reformulation of the original problem.

We contrasted our description of Taguchian RSM with the one in Dellino et al. (2008): the major differences are that we did not assume multivariate normality since we considered stochastic simulation and had a large sample size -so that multivariate normality is the result of a central limit theorem- and we used classic F-test for lack-of-fit.

First, we applied our Taguchian RSM to an inventory example which is used as an introductory exercise to our methodology; and our results showed that low risks mean high costs.

Then we applied our method to a more complex example with three decision variables and a single environmental factor. Based on the results taken from the simulation of the problem in Arena, we applied our algorithms for RSM and lack-of-fit test. Our results show that the presented methodology reaches a global optimum value for the minimization problem.

In order to expand the original example, we did add one more environmental factor and applied our methodology to the simulation results again. Our results show that the presented methodology reaches a local optimum value for the minimization problem for different starting points.

While the author believes that the presented model provides value, there are also further points that can be included. First of all classic RSM is an iterative metaheuristic; hence, in our future work, we should extend our one shot approach to an iterative one. Another issue is to extend the current approach to handle multiple random responses. Additional interactions between and within the decision factors could have been included.

REFERENCES:

[1] Sahinidis, N.V, "Optimization under uncertainty: state-of-the-art and opportunities", *Elsevier*, doi:10.1016/j.compchemeng.2003.09.017, (2004).

[2] Beyer, H.-G., Sendhoff, B., "Robust optimization: a comprehensive survey", *Computer Methods in Applied Mechanics and Engineering,* 196, 3190-3218, (2007).

[3] Shapiro, A., Ruszczyński, A., "Lectures on stochastic programming", http: //www2.isye.gatech.edu/»ashapiro/publications.html, (2008).

[4] Kall, P., Wallace, S.W., *Stochastic Programming*, John Wiley & Sons, Chichester, England, (1994).

[5] Birge, J.R., Louveaux, F., *Introduction to Stochastic Programming*, Springer, (1997).

[6] Ferguson, A., Dantzig, G.B., "The allocation of aircraft to routes: an example of linear programming under uncertain demands", *Management Science*, 3, 45-73, (1956).

[7] Charnes, A., Cooper, W.W., Symonds, G.H., "Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil", *Management Science*, 6, 235-263, (1958).

[8] Dupačová, J., Gairovonski, A., Kos, Z., Szantai, T., "Stochastic programming in water management: a case study and a comparison of solution techniques", *European Journal of Operational Research*, 52, 28-44, (1991).

[9] Manne, A.S., "Waiting for the breeder", in *Review of Economic Studies Symposium*, 47-65, (1974).

[10] Louveaux, F.V., "A solution method for multistage stochastic programs with recourse with application to an energy investment problem", *Operations Research,* 28, 889-902, (1980).

[11] Pereira, M.V.F., Pinto, L.M.V.G., "Multi-stage stochastic optimization applied to energy planning", *Mathematical Programming*, 37, 131-152, (1991).

[12] Manne, A.S., Richels, R.G., *Buying Greenhouse Insurance - The Economic Costs of CO2 Emission Limits*, Cambridge, The MIT Press, (1992).

[13] Morton, D.P., "An enhanced decomposition algorithm for multistage stochastic hydroelectric scheduling", *Annals of Operations Research*, 64, 211-235, (1996).

[14] Takriti, S., Birge, J.R., Long, E., "A stochastic model for the unit commitment problem", *IEEE Transactions on Power Systems*, 11, 1497-1508, (1996).

[15] Carøe, C.C., Ruszczyński, A., Schultz, R., "Unit commitment under uncertainty via two-stage stochastic programming", In D. Pisinger, C.C. Carøe, & J.M. Rygaard (Eds.), *Proceedings of NOAS 97* (pp. 21-30), Department of Computer Science, University of Copenhagen, Denmark, (1997).

[16] Gassmann, H.I., "Optimal harvest of a forest in the precence of uncertainty", *Canadian Journal of Forest Research*, 19, 1267-1274, (1989).

[17] Kao, E.P.C., Queyranne, M., "Budgeting costs of nursing in a hospital", *Management Science*, 31, 608-621, (1985).

[18] Mulvey, J.M., Vladimirou, H., "Stochastic network optimization models for investment planning", *Annals of Operations Research*, 20, 187-217, (1989).

[19] Mulvey, J.M., Vladimirou, H., "Applying the progressive hedging algorithm to stochastic generalized networks", *Annals of Operations Research*, Vol. 31, pp. 399-424, (1991b).

[20] Mulvey, J.M., Vladimirou, H., "Stochastic network programming for financial planning problems", *Management Science*, 38, 1642-1664, (1992).

[21] Ziemba, W.T., Vickson, R.G., Eds., *Stochastic Optimization Models in Finance*, Academic Press, (1975).

[22] Kallberg, J.G., White, R., Ziemba, W.T., "Short Term Financial Planning Under Uncertainty", *Management Science,* (1982).

[23] Zenios, S.A., "Parallel Monte Carlo simulation of mortgage backed securities", *Financial Optimization*, Cambridge University Press, pp. 325-343, (1992).

[24] Dert, C.L., *Asset liability management for pension funds: a multistage chace constrained programming approach*. Ph.D. Thesis, Erasmus University, Rotterdam, The Netherlands, (1995).

[25] Carino, D.R., Ziemba, W.T., "Formulation of the Russel-Yasuda Kasai financial planning model", *Operations Research*, 46, 433-449, (1998).

[26] Kouwenberg, R., Zenios, S.A., *Stochastic programming models for asset liability management*. Technical report, HERMES Center of Excellence on Computational Finance & Economics, University of Cyprus, Nicosia, Cyprus, (2001).

[27] Sherali, H.D., Soyster, A.L., Murphy, F.H., Sen, S., "Allocation of capital costs in electric utility capacity expansion planning under uncertainty", *Management Science,* 30, 1-19, (1984).

[28] Davis, M.H.A., Dempster, M.A.H., Sethi, S.P., Vermes, D., "Optimal capacity expansion under uncertainty", Advances in Applied Probability, 19,156-176, (1987).

[29] Bienstock, D., Shapiro, J.F., "Optimizing resource acquisition decisions by stochastic programming", *Management Science,* 34, 215-229, (1988).

[30] Eppen, G.D., Martin, R.K., Schrage, L., "A scenario approach to capacity planning", *Operations Research*, 37, 517-527, (1989).

[31] Berman, O., Ganz, Z., Wagner, J.M., "A stochastic optimization model for planning capacity expansion in a service industry under uncertain demand", *Naval Research Logistics*, 41, 545-564, (1994).

[32] Malcolm, S., Zenios, S.A., "Robust optimization of power systems capacity expansion under uncertainty", *Journal of the Operational Research Society*, 45, 1040-1049, (1994).

[33] Ahmed, S., King, A., Parija, G.A., "A multi-stage stochastic integer programming approach for capacity expansion under uncertainty", *Journal of Global Optimization*, 26, 3-24, (2003).

[34] Charnes, A., Cooper, W.W., "Chance-constrained programming", *Management Science*, 6, 73-79, (1959).

[35] Charnes, A., Cooper, W.W., "Deterministic equivalents for optimizing and satisfying under chance constraints", *Operations Research*, Vol. 11, 18-39, (1963).

[36] Vajda, S., *Probabilistic Programming*. Academic Press, New York, (1972).

[37] Naslund, B., "A model of capital budgeting under risk", *The Journal of Business*, Vol. 39, 257-271, (1966).

[38] Gurgur, C., Luxhoj, J.T., "Application of chance-constrained programming to capital rationing problems with asymmetrically distributed cash flows and available budget", *The Engineering Economist*, 48(3), 241-258, (2003).

[39] Kelle, P., "Sequential investment planning based on stochastic models", *Engineering Cost and Production Economist*, Vol. 12, 205-209, (1987).

[40] Sarper, H., "Capital rationing under risk: a chance constrained approach using uniformly distributed cash flows and budgets", *The Engineering Economist*, Vol. 39, 49-76, (1993).

[41] Silver, E.A., Pyke, D.F., Peterson, R., *Inventory Management and Production Planning and Scheduling*, Third Edition. Wiley, New York, (1998).

[42] Nemirovski, A., Shapiro, A., "Convex approximations of chance constraint programs", *SIAM Journal on Optimization,* Vol.17, No.4, pp.969-996, (2006).

[43] Artzner, P., Delbaen, F., Eber, J.-M., Heath, D., "Coherent measures of risk", *Mathematical Finance*, 9, 203-228, (1999).

[44] Markowitz, H.M., "Portfolio Selection", *Journal of Finance*, 7, 77-91, (1952).

[45] Takriti, S., Ahmed, S., "On robust optimization of two-stage systems", *Mathematical Programing*, 99, 109-126, (2004).

[46] Ahmed, S., Cakmak, U., Shapiro, A., "Coherent risk measures in inventory problems", *European Journal of Operational Research*, 182 (1), 226-238, (2007).

[47] Rockafellar, R.T., Uryasev, S.P., "Optimization of conditional value-at-risk", *The Journal of Risk*, 2, 21-41, (2000).

[48] Garcia-Gonzalez, J., Parrilla, E., Mateo, A., "Risk-averse profit-based optimal scheduling of a hydro-chain in the day ahead electricity market", *European Journal of Operational Research*, 181, 1354-1369, (2007).

[49] Von Neumann, J., Morgenstern, O., *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, (1947).

[50] Soyster, A.L., "Convex programming with set-inclusive constraints and applications to inexact linear programming", *Operations Research,* 1154-1157, (1973).

[51] Ben-Tal, A., Nemirovski, A., "Robust optimization: methodology and applications", *Mathematical Programming*, Series B, 92, 453-480, (2002).

[52] Taguchi, G., *Quality Engineering through Design Optimization,* Kraus International Publication, New York, (1984).

[53] Trosset, M., "Taguchi and robust optimization", Technical Report 96-31, Department of Computational & Applied Mathematics, Rice University, (1996).

[54] Nair, V.N., Abraham, B., MacKay, J., Box, G., Kacker, R.N., Lorenzen, T.J., Lucas, J.M, Myers, R.H., Vining, G.G., Nelder, J.A., Phadke, M.S., Sacks, J., Welch, W.J., Shoemaker, A.C., Tsui, K.L., Taguchi, S., Wu, C.F.J., "Taguchi's parameter design: a panel discussion", *Technometrics*, 34, 127–161, (1992).

[55] Myers, R.H., Montgomery, D.C., *Response surface methodology: process and product optimization using designed experiments*, Wiley, New York, (1995).

[56] Dellino, G., Kleijnen, J.P.C., Meloni, C., *Robust optimization in simulation: Taguchi and Response Surface Methodology,* Working Paper, Tilburg University, Tilburg, The Netherlands, (2008).

[57] Fu, M.C., "Optimization for simulation: theory vs. practice", *INFORMS Journal on Computing,* 14 192-215, (2002).

[58] Myers, R.H., Carter, W.H., "Response surface techniques for dual response systems", *Technometrics*, 15, 301-317, (1973).

[59] Vining, G.G., Myers, R.H., "Combining Taguchi and response surface philosophies: a dual response approach", *Journal of Quality Technology,* 22, 38-45, (1990).

[60] Fan, S-KS., Del Castillo, E., "Calculation of an optimal region of operation for dual response systems fitted from experimental data", *Journal of the Operational Research Society,* 50, 826-836, (1999).

[61] Yang, T., Kuo, Y., Chou, P., "Solving a multiresponse simulation problem using a dual-response system and scatter search method", *Simulation Modelling Practice and Theory,* 13, 356-369, (2005).

[62] Lee, S.B., Park, C., "Development of robust design optimization using incomplete data", *Computers & Industrial Engineering*, 50, 345-356, (2006).

[63] Köksoy, O., Yalçinöz, T., "Robust design using Pareto type optimization: a genetic algorithm with arithmetic crossover", *Computers & Industrial Engineering,* 55, 208-218, (2008).

[64] Bendell, A., Disney, J., Pridmore, W.A., *Taguchi Methods: Applications in World Industry*, IFS Publications, London, (1987).

[65] Dehnad, K., *Quality Control, Robust Design and the Taguchi Method*, Wadsworth and Brooks/Cole, Pacific Grove, CA, (1989).

[66] Taguchi, G., Wu, Y., *Introduction to Off-Line Quality Control*, Central Japan Quality Control Association: Nagoya (Japan), available from American Supplier Institute, Romulus, MI, U. S. A., (1980).

[67] Taguchi, G., *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Kraus International Publications: White Plains, New York, (1986).

[68] Box, G.E.P., "Signal-to-noise ratios, performance criteria, and transformations", *Technometrics*, 30, 1-17, (1988).

[69] Phadke, S.M., *Quality engineering using robust design*, Englewood Cliffs, NJ, Prentice Hall, (1989).

[70] Welch, W.J., Yu, T-K., Kang, S.M., Sacks, J., "Computer experiments for quality control by parameter design", *Journal of Quality Technology,* 22, 15–22, (1990).

[71] Shoemaker, A.C., Tsui, K.L., Wu, C.F., "Economical experimentation methods for robust design", *Technometrics*, 33, 415-427, (1994).

[72] Pledger, M., "Observable uncontrollable factors in parameter design", *Journal of Quality Technology*, 28, 153-162, (1996).

[73] Borkowski, J.J., Lucas, J.M., "Designs of mixed resolution for process robustness studies", *Technometrics*, 39, 63-70, (1997).

[74] Wu, C.F.J., Hannada, M., *Experiments: Planning, analysis, and parameter design optimization*, John Wiley, New York, (2000).

[75] Lin, P.K.H., Sullivan, L.P., Taguchi, G., "Using Taguchi Methods in quality engineering", *Quality Progress*, Vol 23, N.9, (1990).

[76] Lin, T.Y., Tseng, C.H., "Optimum design for artificial neural networks: an example in a bicycle derailleur system", *Engineering Application of Artificial Intelligence*, 13, 3-14, (2000).

[77] Taguchi, G., Chowdhury, S., Taguchi, S., *Robust Engineering*, McGraw-Hill, New York, (1999).

[78] Wysk, R.A., Niebel, B.W., Cohen, P.H., Simpson, T.W., *Manufacturing Processes: Integrated Product and Process Design*, McGraw-Hill. New York, (2000).

[79] Box, G.E.P., "Discussion of off-line quality control, parameter design and the Taguchi methods", *Journal of Quality Technologies*, 17, 198–206, (1985).

[80] Vining, G.G., Myers, R.H., "Combining Taguchi and response surface philosophies: a dual response approach", *Journal of Quality Technology*, 22, 38-45, (1990).

[81] Pignatiello, J., Ramberg, J.S., "Top ten triumphs and tragedies of Genichi Taguchi", *Quality Engineering*, 4, 221–225, (1991).

[82] Myers, R.H., Khuri, A.I., Vining, G.G., "Response surface alternatives to the Taguchi robust design problem", *Journal of the American Statistical Association,* 46, 131–139, (1992).

[83] Leon, R. V., Shoemaker, A.C., Kackar, R.N., "Performance measures independent of adjustment: an explanation and extension of Taguchi signal-to-noise ratio", *Technometrics*, 29, 253–285, (1987).

[84] Box, G.E.P., Bisgaard, S., Fung, C., "An explanation and critique of Taguchi's contributions to quality engineering", *International Journal of Reliability Management*, 4, 123–131, (1988).

[85] Tsui, K.L., "An overview of Taguchi method and newly developed statistical methods for robust design", *IIE Transactions,* 24, 44–57, (1992).

[86] Tang, L.C., Xu, K.A., "Unified approach for DRS optimization", *Journal of Quality Technology*, 34, 437–447, (2002).

[87] Ross, D.L., Osborne, D.M., George, J.H., "Decision criteria in dual response", *Struct Multidsc Optim*, 23, 460-466, (2002).

[88] Köksoy, O., Doğanaksoy, N., "Joint optimization of mean and standard deviation in response surface experimentation", *Journal of Quality Technology*, 35, 239–252, (2003).

[89] Yang, T., Kuo, Y., Chou, P., "Solving a multiresponse simulation problem using a dual-response system and scatter search method", *Simulation Modelling Practice and Theory,* 13, 356-369, (2005).

[90] Peterson, J.J., Kuhn, A.M., "Ridge analysis with noise variables", *Technometrics,* Vol.47, No.3, (2005).

[91] Jeong, I.-J., Kim, K.-J., Chang, S.Y., "Optimal weighting of bias and variance in DRS optimization", *Journal of Quality Technology,* 37, 3, ABI/INFORM Global pg.236, (2005).

[92] Yeniay, O., Unal, R., Lepsch, R.A., "Using DRSs to reduce variability in launch vehicle design: a case study", *Reliability Enginerring and System Safety*, 91, 407-412, (2006).

[93] Lee, S.B., Park, C., "Development of robust design optimization using incomplete data", *Computers & Industrial Engineering,* 50, 345-356, (2006).

[94] Lee, S.E., Park, C., Cho, B.-R., "Development of a highly efficient and resistant robust design", *International Journal of Production Research,* Vol.45, No.1, 157-167, (2007).

[95] Köksoy, O., Yalçinöz, T., "Robust design using Pareto type optimization: a genetic algorithm with arithmetic crossover", *Computers & Industrial Engineering,* 55, 208-218, (2008).

[96] Quesada, G.M., Del Castillo, E., "A dual response approach to the multivariate robust parameter design problem", *Journal of Quality Technology,* 46, 2, ABI/INFORM Global pg.176, (2004).

[97] Quesada, G.M., Del Castillo, E., "Two approaches for improving the dual response method in robust parameter design", *Journal of Quality Technology,* 36, 2, ABI/INFORM Global pg.154, (2004).

[98] Myers, W.R., Brenneman, W.A., Myers, R.H., "A dual response approach to robust parameter design for a generalized linear model", *Journal of Quality Technology,* 37, 2, ABI/INFORM Global pg.130, (2005).

[99] Rajagopal, R., Del Castillo, E., Peterson, J.J., "Model and distribution-robust process optimization with noise factors", *Journal of Quality Technology,* 37, 3, ABI/INFORM Global pg.210, (2005).

[100] Giovagnoli, A., Romano, D., "Robust design via simulation experiments: a modified DRS approach", *Quality and Reliability Enginerring International,* 24, 401-416, (2008).

[101] Kleijnen, J.P.C., *Design and Analysis of Simulation Experiments*, Springer Science + Business Media, New York, (2008).

[102] Law, A.M., *Simulation Modeling and Analysis*, 4th ed. McGraw-Hill, Boston, (2007).

[103] Theil, H., *Principles of Econometrics*, Wiley, New York, (1971).

[104] Irizarry, M., Kuhl, M.E., Lada, E.K., Subramanian, S., Wilson, J.R., "Analyzing transformation-based simulation metamodels", *IIE Transactions,* 35, 271-283, (2003).

[105] Bashyam, S., Fu, M.C., "Optimization of (s, S) inventory systems with random lead times and a service level constraint", *Management Science,* 44, S243-S256, (1998).

[106] Angun, E., *Black box simulation optimization: generalized response surface methodology*, VDM Verlag, Saarbrucken, Germany, (2008).

[107] Kelton, W.D., Sadowski, R.P., Sadowski, D.A., *Simulation with Arena*, 2nd edn. McGraw-Hill, Boston, (2002).

APPENDIX A:

(s, S) Inventory program simulation code:

```
function [averagecostvector, fillratevector]=inventory_simulation(d1total, d2total, e1total, totalruns, holdingcost, orderingcost, setupcost, numberofperiods, leadtimemean)

%initialization
averagecostvector=[];
fillratevector=[];
for i=1:totalruns
 period=1;
 indicator=0;
 totalcost=0;
 totalsatisfieddemand=0;
 totaldemand=0;
 arrivaltimes=[];
 orders=[];
 orderedarrivaltimes=[];
 orderedorders=[];
 notfullfilledtimes=[];
 notfullfilledorders=[];
 inventorylevel=d2total(i);
 inventoryposition=d2total(i);
 while period <= numberofperiods

%check existing orders
if size(notfullfilledtimes,1) ~= 0
count=0;
```

```
for m=1:size(notfullfilledtimes,1)
            if period == notfullfilledtimes(m,1)
inventorylevel=inventorylevel+notfullfilledorders(m,1);
               count=count+1;
            end
         end
         if count > 0
            newnotfullfilledtimes=[];
            newnotfullfilledorders=[];
            for p=1:(size(notfullfilledtimes,1)-count)
               newnotfullfilledtimes(p,1)=notfullfilledtimes(p+count,1);
               newnotfullfilledorders(p,1)=notfullfilledorders(p+count,1);
            end
            notfullfilledtimes=newnotfullfilledtimes;
            notfullfilledorders=newnotfullfilledorders;
         end
      end
      beforedemandinventorylevel=inventorylevel;

%generate demands
      demand=exprnd(e1total(i));

%update inventory level
      inventorylevel=inventorylevel-demand;

%check inventory position, determine leadtimes
if inventoryposition < d1total(i)
         indicator=1;
         leadtime=poissrnd(leadtimemean);
         arrivaltimes=[arrivaltimes; period+1+leadtime];
         orders=[orders; d2total(i)-inventoryposition];
         [orderedarrivaltimes, index1] = sort(arrivaltimes);
         for k=1:size(index1,1)
```

```
            orderedorders(k,1)=orders(index1(k,1),1);
        end
        notfullfilledtimes=[notfullfilledtimes; period+1+leadtime];
        notfullfilledorders=[notfullfilledorders; d2total(i)-inventoryposition];
        newnotfullfilledtimes=[];
        newnotfullfilledorders=[];
        [newnotfullfilledtimes, index2] = sort(notfullfilledtimes);
        for h=1:size(index2,1)
            newnotfullfilledorders(h,1)=notfullfilledorders(index2(h,1),1);
        end
        notfullfilledtimes=newnotfullfilledtimes;
        notfullfilledorders=newnotfullfilledorders;
    end

%compute cost, total cost, satisfied demand, total satisfied demand, and total demand
cost = max([inventorylevel; 0]) * holdingcost + indicator *(setupcost + (orderingcost *
(d2total(i)-inventoryposition)));
    totalcost=totalcost+cost;
    if beforedemandinventorylevel > 0
        satisfieddemand=min([demand; beforedemandinventorylevel]);
    else
        satisfieddemand=0;
    end
    totalsatisfieddemand=totalsatisfieddemand+satisfieddemand;
    totaldemand=totaldemand+demand;
    inventoryposition=inventorylevel+sum(notfullfilledorders);
    indicator=0;
    period=period + 1;
  end
  averagecostvector=[averagecostvector;totalcost/numberofperiods];
  fillratevector=[fillratevector;totalsatisfieddemand/totaldemand];
end
```

APPENDIX B:

Application of Robust RSM to the inventory simulation:

```
%this program applies robust rsm to an inventory simulation
%fix simulation seed
rand('state',0);
numberofreplicates = 30;
numberofinputs = 13;
totalruns = numberofreplicates * numberofinputs;
holdingcost = 1;
orderingcost = 2;
setupcost = 36;
numberofperiods = 2500;
leadtimemean=6;
alpha=0. 01;
totalreject=0;
totalfailreject=0;
allzetas=[];

%design matrix
d1=[1340; 1340; 1340; 980; 980; 980; 1340; 980; 1160; 1424; 1160; 896; 1160];
d2=[1405; 1405; 1019; 1405; 1019; 1405; 1019; 1019; 1212; 1212; 1476; 1212; 948];
e1=[120; 80; 120; 120; 120; 80; 80; 80; 100; 100; 100; 100; 100];

d1total=[repmat(d1(1),numberofreplicates,1);      repmat(d1(2),numberofreplicates,1);
repmat(d1(3),numberofreplicates,1);               repmat(d1(4),numberofreplicates,1);
repmat(d1(5),numberofreplicates,1);               repmat(d1(6),numberofreplicates,1);
repmat(d1(7),numberofreplicates,1);               repmat(d1(8),numberofreplicates,1);
```

```
repmat(d1(9),numberofreplicates,1);          repmat(d1(10),numberofreplicates,1);
repmat(d1(11),numberofreplicates,1);         repmat(d1(12),numberofreplicates,1);
repmat(d1(13),numberofreplicates,1)];

d2total=[repmat(d2(1),numberofreplicates,1);   repmat(d2(2),numberofreplicates,1);
repmat(d2(3),numberofreplicates,1);            repmat(d2(4),numberofreplicates,1);
repmat(d2(5),numberofreplicates,1);            repmat(d2(6),numberofreplicates,1);
repmat(d2(7),numberofreplicates,1);            repmat(d2(8),numberofreplicates,1);
repmat(d2(9),numberofreplicates,1);           repmat(d2(10),numberofreplicates,1);
repmat(d2(11),numberofreplicates,1);          repmat(d2(12),numberofreplicates,1);
repmat(d2(13),numberofreplicates,1)];

e1total=[repmat(e1(1),numberofreplicates,1);   repmat(e1(2),numberofreplicates,1);
repmat(e1(3),numberofreplicates,1);            repmat(e1(4),numberofreplicates,1);
repmat(e1(5),numberofreplicates,1);            repmat(e1(6),numberofreplicates,1);
repmat(e1(7),numberofreplicates,1);            repmat(e1(8),numberofreplicates,1);
repmat(e1(9),numberofreplicates,1);           repmat(e1(10),numberofreplicates,1);
repmat(e1(11),numberofreplicates,1);          repmat(e1(12),numberofreplicates,1);
repmat(e1(13),numberofreplicates,1)];

X=[ones(totalruns,1) d1total d2total d1total. ^2 d2total. ^2 d1total. *d2total e1total
d1total. *e1total d2total. *e1total];
fid=fopen('zeta','w+');
fprintf(fid,'zeta(1) zeta(2) zeta(3) zeta(4) zeta(5) zeta(6) zeta(7) zeta(8)
zeta(9)\n');
for t=1:50

%call inventory simulation
[averagecostvector, fillratevector]=inventory_simulation(d1total, d2total, e1total,
totalrun, holdingcost, orderingcost, setupcost, numberofperiods, leadtimemean);

%fit regression metamodel to average cost realizations
   zeta=X\averagecostvector;
```

```
%call lack-of-fit F-test
[F_statistic,F_critical]=lack_of_fit(zeta, alpha, X, averagecostvector, numberofreplicates, totalruns, numberofinputs);
   if F_statistic > F_critical
      totalreject=totalreject+1;
   else
      totalfailreject=totalfailreject+1;
   end
   allzetas=[allzetas;zeta(1) zeta(2) zeta(3) zeta(4) zeta(5) zeta(6) zeta(7) zeta(8) zeta(9)];
   fprintf(fid,' %g %g %g %g %g %g %g %g %g\n',zeta(1), zeta(2), zeta(3), zeta(4), zeta(5), zeta(6), zeta(7), zeta(8), zeta(9));
   t
end
fclose(fid);
```

APPENDIX C:

Lack-of-fit test simulation code:

```
%this function performs lack-of-fit F-test
function [F_statistic, F_critical]=lack_of_fit(zeta, alpha, X, averagecostvector, numberofreplicates, totalruns, numberofinputs)
one=[]; two=[]; three=[]; four=[]; five=[]; six=[]; seven=[]; eight=[]; nine=[]; ten=[]; eleven=[]; twelve=[]; thirteen=[];
averagecostvectorbar=[];

%compute sum of squared residuals and sum of squared errors
regressionhead=X*zeta;
sumsquaredresidual=(averagecostvector-regressionhead)'*(averagecostvector-regressionhead);
for i=1:totalruns
   if i<= numberofreplicates
      one=[one;averagecostvector(i)];
   elseif (i>= numberofreplicates+1)&(i<= 2*numberofreplicates)
      two=[two;averagecostvector(i)];
   elseif (i>= 2*numberofreplicates+1)&(i<= 3*numberofreplicates)
      three=[three;averagecostvector(i)];
   elseif (i>= 3*numberofreplicates+1)&(i<= 4*numberofreplicates)
      four=[four;averagecostvector(i)];
   elseif (i>= 4*numberofreplicates+1)&(i<= 5*numberofreplicates)
      five=[five;averagecostvector(i)];
   elseif (i>= 5*numberofreplicates+1)&(i<= 6*numberofreplicates)
      six=[six;averagecostvector(i)];
   elseif (i>= 6*numberofreplicates+1)&(i<= 7*numberofreplicates)
```

```
        seven=[seven;averagecostvector(i)];
    elseif (i>= 7*numberofreplicates+1)&(i<= 8*numberofreplicates)
        eight=[eight;averagecostvector(i)];
    elseif (i>= 8*numberofreplicates+1)&(i<= 9*numberofreplicates)
        nine=[nine;averagecostvector(i)];
    elseif (i>= 9*numberofreplicates+1)&(i<= 10*numberofreplicates)
        ten=[ten;averagecostvector(i)];
    elseif (i>= 10*numberofreplicates+1)&(i<= 11*numberofreplicates)
        eleven=[eleven;averagecostvector(i)];
    elseif (i>= 11*numberofreplicates+1)&(i<= 12*numberofreplicates)
        twelve=[twelve;averagecostvector(i)];
    elseif (i>= 12*numberofreplicates+1)&(i<= 13*numberofreplicates)
        thirteen=[thirteen;averagecostvector(i)];
    end
end
for i=1:totalruns
    if i<= numberofreplicates
        averagecostvectorbar=[averagecostvectorbar;mean(one)];
    elseif (i>= numberofreplicates+1)&(i<= 2*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(two)];
    elseif (i>= 2*numberofreplicates+1)&(i<= 3*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(three)];
    elseif (i>= 3*numberofreplicates+1)&(i<= 4*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(four)];
    elseif (i>= 4*numberofreplicates+1)&(i<= 5*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(five)];
    elseif (i>= 5*numberofreplicates+1)&(i<= 6*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(six)];
    elseif (i>= 6*numberofreplicates+1)&(i<= 7*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(seven)];
    elseif (i>= 7*numberofreplicates+1)&(i<= 8*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(eight)];
```

```
    elseif (i>= 8*numberofreplicates+1)&(i<= 9*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(nine)];
    elseif (i>= 9*numberofreplicates+1)&(i<= 10*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(ten)];
    elseif (i>= 10*numberofreplicates+1)&(i<= 11*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(eleven)];
    elseif (i>= 11*numberofreplicates+1)&(i<= 12*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(twelve)];
    elseif (i>= 12*numberofreplicates+1)&(i<= 13*numberofreplicates)
        averagecostvectorbar=[averagecostvectorbar;mean(thirteen)];
    end
end
sumsquaredpurerror=(averagecostvector-averagecostvectorbar)'*(averagecostvector-
averagecostvectorbar);
sumsquaredlackfit=sumsquaredresidual-sumsquaredpurerror;

%perform F-test
dof1=numberofinputs-size(zeta,1);
dof2=totalruns-numberofinputs;
F_statistic=(sumsquaredlackfit/dof1)/(sumsquaredpurerror/dof2);
F_critical = finv(1-alpha,dof1,dof2);
```

Objective function @ *fmincon*

```
%this function creates the objective function for the minimization
function obj = objective(x)
obj = -180.0543 + 0.1704*x(1) + 0.4675*x(2) - 0.0004*x(1)^2 - 0.0003*x(2)^2 +
0.001*x(1)*x(2) + 2.58*x(3) - 0.0019*x(1)*x(3) - 0.0035*x(2)*x(3);
```

Minimization function

```
%this program minimizes the regression predictor subject to its variance <=
%target value where target value takes its values from [150, 400]
[x,fval,exitflag]=fmincon(@objective,[1160;1212;100], [1 -1 0], 0, [], [], [896;948;80],
[1424;1476;120], @constraint)
```

APPENDIX D:

Table D.1.: The percentages of requests for transfer options and product types

3 transfer options choice percentages	
Technical support	76%
Sales information	16%
Order status inquiry	8%
3 product type options choice percentages	
Product type 1	25%
Product type 2	34%
Product type 3	41%

Table D.2.: Sales staff (7) daily schedules (number of people @ time period in minutes)

Sales staff 1	3@90
Sales staff 2	7@90
Sales staff 3	6@90
Sales staff 4	7@60
Sales staff 5	6@120
Sales staff 6	7@120
Sales staff 7	4@90

Table D.3.: Estimated times for: (in minutes)

Support option choice time	UNIF (0.1,0.6)
Technical support product type choice	UNIF (0.1,0.5)
All technical support calls	TRIA(3,6,18)
Response preparation time for further investigation required technical calls	EXPO(60)
Customer recall time	TRIA(2,4,9)
Sales calls	TRIA(4,15,45)
Order status call transactions	TRIA(2,3,4)
Follow up order status calls	TRIA(3,5,10)

Table D.4.: Call Arrival Rates (Calls Per Hour)

Time	Rate	Time	Rate	Time	Rate	Time	Rate
8.00-8.30	20	10.30-11.00	75	13.00-13.30	110	15.30-16.00	90
8.30-9.00	35	11.00-11.30	75	13.30-14.00	95	16.00-16.30	70
9.00-9.30	45	11.30-12.00	90	14.00-14.30	105	16.30-17.00	65
9.30-10.00	50	12.00-12.30	95	14.30-15.00	90	17.00-17.30	45
10.00-10.30	70	12.30-13.00	105	15.00-15.30	85	17.30-18.00	30

Table D.5.: Technical support (11) schedules

Name	Product lines	Time Period (30 minutes)																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Tech1	1	·	·	·	·	·	·	·		·	·	·	·	·	·	·	·	·					
Tech2	1						·	·	·	·	·	·		·	·	·	·	·	·	·	·	·	·
Tech3	1,3			·	·	·	·	·	·		·	·	·	·	·	·	·	·	·	·			
Tech4	1,2,3					·	·	·	·	·		·	·	·	·	·	·	·	·	·	·	·	
Tech5	1,2,3				·	·	·	·	·	·	·		·	·	·	·	·	·	·	·	·		
Tech6	2	·	·	·	·	·	·	·		·	·	·	·	·	·	·	·	·					
Tech7	2						·	·	·	·		·	·	·	·	·	·	·	·	·	·	·	·
Tech8	2				·	·	·	·	·		·	·	·	·	·	·	·	·	·	·	·		
Tech9	3	·	·	·	·	·	·	·		·	·	·	·	·	·	·	·	·					
Tech10	3						·	·	·	·		·	·	·	·	·	·	·	·	·	·	·	·
Tech11	3				·	·	·	·	·		·	·	·	·	·	·	·	·	·	·	·		

Table D.6.: Calculation of the central point, positive and negative axial points and positive and negative values of design variables based on *OptQuest* Results

OptQuest Results	$\alpha = 1\%$		$radius = \sqrt{(26.26-26)^2 + (4.04-4)^2 + (5.05-5)^2 + (0.505-0.5)^2} = 0.2678$	
	+	-	positive axial points	negative axial points
$x_1^* = 26$	26.26	25.74	$x_1 = 26.2678,\ x_2 = 4,\ x_3 = 5,\ e_1 = 0.5$	$x_1 = 25.7321,\ x_2 = 4,\ x_3 = 5,\ e_1 = 0.5$
$x_2^* = 4$	4.04	3.96	$x_1 = 26,\ x_2 = 4.2678,\ x_3 = 5,\ e_1 = 0.5$	$x_1 = 26,\ x_2 = 3.7321,\ x_3 = 5,\ e_1 = 0.5$
$x_3^* = 5$	5.05	4.95	$x_1 = 26,\ x_2 = 4,\ x_3 = 5.2678,\ e_1 = 0.5$	$x_1 = 26,\ x_2 = 4,\ x_3 = 4.7321,\ e_1 = 0.5$
$e_1^* = 0.5$	0.505	0.495	central point	
			$x_1 = 26,\ x_2 = 4,\ x_3 = 5,\ e_1 = 0.5$	

Table D.7.: Input combinations table for the call centre simulation

23 input combinations								
2^4 input combinations Resolution V Design								
x_1	x_2	x_3	e_1		x_1	x_2	x_3	e_1
+	+	+	+		26.26	4.04	5.05	0.505
+	+	+	-		26.26	4.04	5.05	0.495
+	+	-	+		26.26	4.04	4.95	0.505
+	-	+	+		26.26	3.96	5.05	0.505
-	+	+	+		25.74	4.04	5.05	0.505
-	-	+	+		25.74	3.96	5.05	0.505
-	+	-	+		25.74	4.04	4.95	0.505
-	+	+	-		25.74	4.04	5.05	0.495
+	-	-	+		26.26	3.96	4.95	0.505
+	+	-	-		26.26	4.04	4.95	0.495
+	-	+	-		26.26	3.96	5.05	0.495
-	-	-	+		25.74	3.96	4.95	0.505
-	-	+	-		25.74	3.96	5.05	0.495
-	+	-	-		25.74	4.04	4.95	0.495
+	-	-	-		26.26	3.96	4.95	0.495
-	-	-	-		25.74	3.96	4.95	0.495
central point					26	4	5	0.5
positive axial points					26.2678	4	5	0.5
					26	4.2678	5	0.5
					26	4	5.2678	0.5
negative axial points					25.7321	4	5	0.5
					26	3.7321	5	0.5
					26	4	4.7321	0.5

Table D.8.: Total sytem costs per replication calculated through the call centre simulation

Input combinations	Total sytem cost @*l*th replication										Total cost
	1	2	3	4	5	6	7	8	9	10	
i.c. 1	2974.50	3249.27	4351.07	3819.31	3866.04	4034.41	4158.34	4283.42	4847.41	3221.92	3881
i.c. 2	2144.48	4734.73	3726.44	4435.67	3571.43	4242.04	4558.35	4088.35	3849.73	4141.91	3949
i.c. 3	2910.36	3054.69	4079.72	4534.55	3651.75	4263.69	4391.89	3905.11	4223.16	3978.76	3899
i.c. 4	2968.74	3243.51	4345.31	3813.55	3860.28	4028.65	4152.58	4277.66	4841.65	3216.16	3875
i.c. 5	2906.60	3894.78	3922.27	4162.51	4271.60	4207.29	3949.10	4403.57	3117.10	4357.22	3919
i.c. 6	2900.84	3889.02	3916.51	4156.75	4265.84	4201.53	3943.34	4397.81	3111.34	4351.46	3913
i.c. 7	2971.87	3180.87	4052.82	3987.83	4506.67	4432.59	3121.14	4823.22	3266.66	3931.37	3828
i.c. 8	2144.48	4459.83	4009.44	4342.30	3960.66	4221.37	3760.86	4086.63	4105.72	4200.27	3929
i.c. 9	2904.60	3048.93	4073.96	4528.79	3645.99	4257.93	4386.13	3899.35	4217.40	3973.00	3894
i.c. 10	2205.68	4323.85	3968.36	3746.25	4579.18	4010.62	4151.34	4124.97	4284.05	3559.01	3895
i.c. 11	2138.72	4728.97	3720.68	4429.91	3565.67	4236.28	4552.59	4082.59	3843.97	4136.15	3944
i.c. 12	2966.11	3175.11	4047.06	3982.07	4500.91	4426.83	3115.38	4817.46	3260.90	3925.61	3822
i.c. 13	2132.72	4454.07	4003.68	4336.54	3954.90	4215.61	3755.10	4080.87	4099.96	4194.51	3923
i.c. 14	2205.68	4306.03	4403.08	3692.27	3970.70	4224.98	3875.20	3804.03	4165.76	3450.93	3810
i.c. 15	2199.92	4318.09	3962.60	3740.49	4573.42	4004.86	4145.58	4119.21	4278.29	3553.25	3890
i.c. 16	2199.92	4300.27	4397.32	3686.51	3964.94	4219.22	3869.44	3798.27	4160.00	3445.17	3804
centeral point	2817.39	3902.39	3805.63	4225.54	4138.06	4088.09	4288.14	4047.90	4277.02	3839.19	3943
positive axial point 1	2817.39	3902.39	3805.63	4225.54	4138.06	4088.09	4288.14	4047.90	4277.02	3839.19	3943
positive axial point 2	2836.68	3921.67	3824.91	4244.82	4157.35	4107.37	4307.42	4067.18	4296.30	3858.47	3962
positive axial point 3	2835.60	3920.60	3823.84	4243.75	4156.28	4106.30	4306.35	4066.11	4295.23	3857.40	3961
negative axial point 1	2492.14	3703.68	4397.66	4521.43	4493.66	3897.11	4301.80	2818.23	4740.95	4730.45	4010
negative axial point 2	2798.10	3883.10	3786.34	4206.25	4118.78	4068.80	4268.85	4028.61	4257.73	3819.90	3924
negative axial point 3	2867.18	3750.88	4215.76	3256.36	4664.11	3737.27	3814.78	4596.91	3330.91	4740.62	3897

APPENDIX E:

```
%this function performs lack-of-fit F-test
function [F_statistic, F_critical]=lack_of_fit2(beta, alpha, X, costvectorperreplications,
averagecost vector, numberofreplications, totalruns, numberofinputs)

%compute sum of squared residuals and sum of squared errors
averagecostvectorbar=[];
regressionhead=X*beta;
sumsquaredresidual=(costvectorperreplications-
      regressionhead)'*(costvectorperreplications-regressionhead);
for i=1:totalruns
   if i<= numberofreplications
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(1)];
   elseif (i>= numberofreplications+1)&&(i<= 2*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(2)];
   elseif (i>= 2*numberofreplications+1)&&(i<= 3*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(3)];
   elseif (i>= 3*numberofreplications+1)&&(i<= 4*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(4)];
   elseif (i>= 4*numberofreplications+1)&&(i<= 5*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(5)];
   elseif (i>= 5*numberofreplications+1)&&(i<= 6*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(6)];
   elseif (i>= 6*numberofreplications+1)&&(i<= 7*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(7)];
   elseif (i>= 7*numberofreplications+1)&&(i<= 8*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(8)];
```

```
    elseif (i>= 8*numberofreplications+1)&&(i<= 9*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(9)];
    elseif (i>= 9*numberofreplications+1)&&(i<= 10*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(10)];
    elseif (i>= 10*numberofreplications+1)&&(i<= 11*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(11)];
    elseif (i>= 11*numberofreplications+1)&&(i<= 12*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(12)];
    elseif (i>= 12*numberofreplications+1)&&(i<= 13*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(13)];
    elseif (i>= 13*numberofreplications+1)&&(i<= 14*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(14)];
    elseif (i>= 14*numberofreplications+1)&&(i<= 15*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(15)];
    elseif (i>= 15*numberofreplications+1)&&(i<= 16*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(16)];
    elseif (i>= 16*numberofreplications+1)&&(i<= 17*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(17)];
    elseif (i>= 17*numberofreplications+1)&&(i<= 18*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(18)];
    elseif (i>= 18*numberofreplications+1)&&(i<= 19*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(19)];
    elseif (i>= 19*numberofreplications+1)&&(i<= 20*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(20)];
    elseif (i>= 20*numberofreplications+1)&&(i<= 21*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(21)];
    elseif (i>= 21*numberofreplications+1)&&(i<= 22*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(22)];
    elseif (i>= 22*numberofreplications+1)&&(i<= 23*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(23)];
    end
end
```

```
sumsquaredpurerror=(costvectorperreplications-
        averagecostvectorbar)'*(costvectorperreplications-averagecostvectorbar);
sumsquaredlackfit=sumsquaredresidual-sumsquaredpurerror;

%perform F-test
dof1=numberofinputs-size(beta,1);
dof2=totalruns-numberofinputs;
F_statistic=(sumsquaredlackfit/dof1)/(sumsquaredpurerror/dof2);
F_critical = finv(1-alpha,dof1,dof2);
```

APPENDIX F:

```
%this program applies robust rsm to the call center simulation
numberofinputs = 23;
numberofreplications=10;
totalruns = numberofinputs*numberofreplications;
alpha=0.01;
totalreject=0;
totalfailreject=0;
allbetas=[];

%design matrix
d1=[26.26; 26.26; 26.26; 26.26; 25.74; 25.74; 25.74; 25.74; 26.26; 26.26; 26.26; 25.74;
25.74; 25.74; 26.26; 25.74; 26; 26.2678; 26; 26; 25.7321; 26; 26];
d2=[4.04; 4.04; 4.04; 3.96; 4.04; 3.96; 4.04; 4.04; 3.96; 4.04; 3.96; 3.96; 3.96; 4.04;
3.96; 3.96; 4; 4; 4.2678; 4; 4; 3.7321; 4];
d3=[5.05; 5.05; 4.95; 5.05; 5.05; 5.05; 4.95; 5.05; 4.95; 4.95; 5.05; 4.95; 5.05; 4.95;
4.95; 4.95; 5; 5; 5; 5.2678; 5; 5; 4.7321];
e1=[0.505; 0.495; 0.505; 0.505; 0.505; 0.505; 0.505; 0.495; 0.505; 0.495; 0.495; 0.505;
0.495; 0.495; 0.495; 0.495; 0.5; 0.5; 0.5; 0.5; 0.5; 0.5; 0.5];

d1total=[repmat(d1(1),numberofreplications,1);   repmat(d1(2),numberofreplications,1);
repmat(d1(3),numberofreplications,1);            repmat(d1(4),numberofreplications,1);
repmat(d1(5),numberofreplications,1);            repmat(d1(6),numberofreplications,1);
repmat(d1(7),numberofreplications,1);            repmat(d1(8),numberofreplications,1);
repmat(d1(9),numberofreplications,1);           repmat(d1(10),numberofreplications,1);
repmat(d1(11),numberofreplications,1);          repmat(d1(12),numberofreplications,1);
repmat(d1(13),numberofreplications,1);          repmat(d1(14),numberofreplications,1);
repmat(d1(15),numberofreplications,1);          repmat(d1(16),numberofreplications,1);
```

```
repmat(d1(17),numberofreplications,1);          repmat(d1(18),numberofreplications,1);
repmat(d1(19),numberofreplications,1);          repmat(d1(20),numberofreplications,1);
repmat(d1(21),numberofreplications,1);          repmat(d1(22),numberofreplications,1);
repmat(d1(23),numberofreplications,1)];

d2total=[repmat(d2(1),numberofreplications,1);  repmat(d2(2),numberofreplications,1);
repmat(d2(3),numberofreplications,1);           repmat(d2(4),numberofreplications,1);
repmat(d2(5),numberofreplications,1);           repmat(d2(6),numberofreplications,1);
repmat(d2(7),numberofreplications,1);           repmat(d2(8),numberofreplications,1);
repmat(d2(9),numberofreplications,1);          repmat(d2(10),numberofreplications,1);
repmat(d2(11),numberofreplications,1);         repmat(d2(12),numberofreplications,1);
repmat(d2(13),numberofreplications,1);         repmat(d2(14),numberofreplications,1);
repmat(d2(15),numberofreplications,1);         repmat(d2(16),numberofreplications,1);
repmat(d2(17),numberofreplications,1);         repmat(d2(18),numberofreplications,1);
repmat(d2(19),numberofreplications,1);         repmat(d2(20),numberofreplications,1);
repmat(d2(21),numberofreplications,1);         repmat(d2(22),numberofreplications,1);
repmat(d2(23),numberofreplications,1)];

d3total=[repmat(d3(1),numberofreplications,1);  repmat(d3(2),numberofreplications,1);
repmat(d3(3),numberofreplications,1);           repmat(d3(4),numberofreplications,1);
repmat(d3(5),numberofreplications,1);           repmat(d3(6),numberofreplications,1);
repmat(d3(7),numberofreplications,1);           repmat(d3(8),numberofreplications,1);
repmat(d3(9),numberofreplications,1);          repmat(d3(10),numberofreplications,1);
repmat(d3(11),numberofreplications,1);         repmat(d3(12),numberofreplications,1);
repmat(d3(13),numberofreplications,1);         repmat(d3(14),numberofreplications,1);
repmat(d3(15),numberofreplications,1);         repmat(d3(16),numberofreplications,1);
repmat(d3(17),numberofreplications,1);         repmat(d3(18),numberofreplications,1);
repmat(d3(19),numberofreplications,1);         repmat(d3(20),numberofreplications,1);
repmat(d3(21),numberofreplications,1);         repmat(d3(22),numberofreplications,1);
repmat(d3(23),numberofreplications,1)];

e1total=[repmat(e1(1),numberofreplications,1);  repmat(e1(2),numberofreplications,1);
repmat(e1(3),numberofreplications,1);           repmat(e1(4),numberofreplications,1);
```

```
repmat(e1(5),numberofreplications,1);        repmat(e1(6),numberofreplications,1);
repmat(e1(7),numberofreplications,1);        repmat(e1(8),numberofreplications,1);
repmat(e1(9),numberofreplications,1);        repmat(e1(10),numberofreplications,1);
repmat(e1(11),numberofreplications,1);       repmat(e1(12),numberofreplications,1);
repmat(e1(13),numberofreplications,1);       repmat(e1(14),numberofreplications,1);
repmat(e1(15),numberofreplications,1);       repmat(e1(16),numberofreplications,1);
repmat(e1(17),numberofreplications,1);       repmat(e1(18),numberofreplications,1);
repmat(e1(19),numberofreplications,1);       repmat(e1(20),numberofreplications,1);
repmat(e1(21),numberofreplications,1);       repmat(e1(22),numberofreplications,1);
repmat(e1(23),numberofreplications,1)];

X=[ones(totalruns,1) d1total d2total d3total e1total d1total.*d2total d1total.*d3total
d2total.*d3total d1total.*e1total d2total.*e1total d3total.*e1total d1total.^2 d2total.^2
d3total.^2];
fid=fopen('call_center','w+');
fprintf(fid,'beta(1) beta(2) beta(3) beta(4) beta(5) beta(6) beta(7) beta(8)
beta(9)        beta(10) beta(11) beta(12) beta(13) beta(14)\n');
for t=1:50

%230*1 cost vector
costvectorperreplications=[2974.50; 3249.27; 4351.07; 3819.31; 3866.04; 4034.41;
4158.34; 4283.42; 4847.41; 3221.92; 2144.48; 4734.73; 3726.44; 4435.67; 3571.43;
4242.04; 4558.35; 4088.35; 3849.73; 4141.91; 2910.36; 3054.69; 4079.72; 4534.55;
3651.75; 4263.69; 4391.89; 3905.11; 4223.16; 3978.76; 2968.74; 3243.51; 4345.31;
3813.55; 3860.28; 4028.65; 4152.58; 4277.66; 4841.65; 3216.16; 2906.60; 3894.78;
3922.27; 4162.51; 4271.60; 4207.29; 3949.10; 4403.57; 3117.10; 4357.22; 2900.84;
3889.02; 3916.51; 4156.75; 4265.84; 4201.53; 3943.34; 4397.81; 3111.34; 4351.46;
2971.87; 3180.87; 4052.82; 3987.83; 4506.67; 4432.59; 3121.14; 4823.22; 3266.66;
3931.37; 2144.48; 4459.83; 4009.44; 4342.30; 3960.66; 4221.37; 3760.86; 4086.63;
4105.72; 4200.27; 2904.60; 3048.93; 4073.96; 4528.79; 3645.99; 4257.93; 4386.13;
3899.35; 4217.40; 3973.00; 2205.68; 4323.85; 3968.36; 3746.25; 4579.18; 4010.62;
4151.34; 4124.97; 4284.05; 3559.01; 2138.72; 4728.97; 3720.68; 4429.91; 3565.67;
4236.28; 4552.59; 4082.59; 3843.97; 4136.15; 2966.11; 3175.11; 4047.06; 3982.07;
```

```
4500.91; 4426.83; 3115.38; 4817.46; 3260.90; 3925.61; 2132.72; 4454.07; 4003.68;
4336.54; 3954.90; 4215.61; 3755.10; 4080.87; 4099.96; 4194.51; 2205.68; 4306.03;
4403.08; 3692.27; 3970.70; 4224.98; 3875.20; 3804.03; 4165.76; 3450.93; 2199.92;
4318.09; 3962.60; 3740.49; 4573.42; 4004.86; 4145.58; 4119.21; 4278.29; 3553.25;
2199.92; 4300.27; 4397.32; 3686.51; 3964.94; 4219.22; 3869.44; 3798.27; 4160.00;
3445.17; 2817.39; 3902.39; 3805.63; 4225.54; 4138.06; 4088.09; 4288.14; 4047.90;
4277.02; 3839.19; 2817.39; 3902.39; 3805.63; 4225.54; 4138.06; 4088.09; 4288.14;
4047.90; 4277.02; 3839.19; 2836.68; 3921.67; 3824.91; 4244.82; 4157.35; 4107.37;
4307.42; 4067.18; 4296.30; 3858.47; 2835.60; 3920.60; 3823.84; 4243.75; 4156.28;
4106.30; 4306.35; 4066.11; 4295.23; 3857.40; 2492.14; 3703.68; 4397.66; 4521.43;
4493.66; 3897.11; 4301.80; 2818.23; 4740.95; 4730.45; 2798.10; 3883.10; 3786.34;
4206.25; 4118.78; 4068.80; 4268.85; 4028.61; 4257.73; 3819.90; 2867.18; 3750.88;
4215.76; 3256.36; 4664.11; 3737.27; 3814.78; 4596.91; 3330.91; 4740.62];

%23*1 vector
averagecostvector=[3881; 3949; 3899; 3875; 3919; 3913; 3828; 3929; 3894; 3895;
3944; 3822; 3923; 3810; 3890; 3804; 3943; 3943; 3962; 3961; 4010; 3924; 3897];

%fit regression metamodel to average cost realizations
   beta=X\costvectorperreplications;

%call lack-of-fit F-test
[F_statistic,F_critical]=lack_of_fit2(beta,alpha,X,costvectorperreplications,
averagecostvector,numberofreplications,totalruns,numberofinputs);
   if F_statistic > F_critical
      totalreject=totalreject+1;
   else
      totalfailreject=totalfailreject+1;
   end
allbetas=[allbetas;beta(1) beta(2) beta(3) beta(4) beta(5) beta(6) beta(7) beta(8) beta(9)
beta(10) beta(11) beta(12) beta(13) beta(14)];
```

```
fprintf(fid,' %g %g %g %g %g %g %g %g %g\n',beta(1), beta(2), beta(3), beta(4), beta(5), beta(6), beta(7), beta(8), beta(9), beta(10), beta(11), beta(12), beta(13), beta(14));
   t
end
fclose(fid);
```

Objective function @ *fmincon*

```
%this function creates the objective function for the minimization
function obj = objective2(x)
obj = -894160 + 46944.3*x(1) + 363.787*x(2) + 71341.3*x(3) + 431242*x(4) - 3.60577*x(1)*x(2) - 1687.57*x(1)*x(3) + 18.75*x(2)*x(3) - 6989.71*x(4)*x(1) - 187.5*x(4)*x(2) - 50035.5*x(4)*x(3) -672.168*x(1)^2 - 24.7372*x(2)^2 - 229.493*x(3)^2;
```

Minimization function

```
[x,fval,exitflag]=fmincon(@objective2, [26;4;5;0.5], [0 1 1 0], 15, [], [], [26;0;0;0.495], [50;15;15;0.505])
```

APPENDIX G:

Table G.1.: Calculation of the central point, positive and negative axial points and positive and negative values of design variables based on *OptQuest* Results

OptQuest Results	$\alpha = 1\%$		$radius = \sqrt{(26.26-26)^2 + (4.04-4)^2 + (5.05-5)^2 + (0.505-0.5)^2 + (0.606-0.6)^2} = 0.267882$	
	+	-	positive axial points	negative axial points
$x_1^* = 26$	26.26	25.74	$x_1 = 26.267882,\ x_2 = 4,\ x_3 = 5,\ e_1 = 0.5,\ e_2 = 0.6$	$x_1 = 25.732118,\ x_2 = 4,\ x_3 = 5,\ e_1 = 0.5,\ e_2 = 0.6$
$x_2^* = 4$	4.04	3.96	$x_1 = 26,\ x_2 = 4.267882,\ x_3 = 5,\ e_1 = 0.5,\ e_2 = 0.6$	$x_1 = 26,\ x_2 = 3.732118,\ x_3 = 5,\ e_1 = 0.5,\ e_2 = 0.6$
$x_3^* = 5$	5.05	4.95	$x_1 = 26,\ x_2 = 4,\ x_3 = 5.267882,\ e_1 = 0.5,\ e_2 = 0.6$	$x_1 = 26,\ x_2 = 4,\ x_3 = 4.732118,\ e_1 = 0.5,\ e_2 = 0.6$
$e_1^* = 0.5$	0.505	0.495	central point	
$e_2^* = 0.6$	0.606	0.594	$x_1 = 26,\ x_2 = 4,\ x_3 = 5,\ e_1 = 0.5,\ e_2 = 0.6$	

Table G.2.: Input combinations table for the call centre simulation

39 input combinations (@application #2)									
2^5 input combinations Resolution V Design									
x_1	x_2	x_3	e_1	e_2	x_1	x_2	x_3	e_1	e_2
+	+	+	+	+	26.26	4.04	5.05	0.505	0.606
+	+	+	+	-	26.26	4.04	5.05	0.505	0.594
+	+	+	-	+	26.26	4.04	5.05	0.495	0.606
+	+	-	+	+	26.26	4.04	4.95	0.505	0.606
+	-	+	+	+	26.26	3.96	5.05	0.505	0.606
-	+	+	+	+	25.74	4.04	5.05	0.505	0.606
-	-	+	+	+	25.74	3.96	5.05	0.505	0.606
-	+	-	+	+	25.74	4.04	4.95	0.505	0.606
-	+	+	-	+	25.74	4.04	5.05	0.495	0.606
-	+	+	+	-	25.74	4.04	5.05	0.505	0.594
+	-	-	+	+	26.26	3.96	4.95	0.505	0.606
+	+	-	-	+	26.26	4.04	4.95	0.495	0.606
+	+	+	-	-	26.26	4.04	5.05	0.495	0.594
+	-	+	-	+	26.26	3.96	5.05	0.495	0.606
+	-	+	+	-	26.26	3.96	5.05	0.505	0.594
+	+	-	+	-	26.26	4.04	4.95	0.505	0.594
-	-	-	+	+	25.74	3.96	4.95	0.505	0.606
+	+	-	-	-	26.26	4.04	4.95	0.495	0.594
-	-	+	-	+	25.74	3.96	5.05	0.495	0.606
-	+	-	-	+	25.74	4.04	4.95	0.495	0.606
-	+	+	-	-	25.74	4.04	5.05	0.495	0.594
+	-	-	+	-	26.26	3.96	4.95	0.505	0.594
+	-	+	-	-	26.26	3.96	5.05	0.495	0.594
-	-	+	+	-	25.74	3.96	5.05	0.505	0.594
-	+	-	+	-	25.74	4.04	4.95	0.505	0.594
+	-	-	-	+	26.26	3.96	4.95	0.495	0.606
-	-	-	-	+	25.74	3.96	4.95	0.495	0.606
+	-	-	-	-	26.26	3.96	4.95	0.495	0.594
-	+	-	-	-	25.74	4.04	4.95	0.495	0.594

-	-	+	-	-	25.74	3.96	5.05	0.495	0.594
-	-	-	+	-	25.74	3.96	4.95	0.505	0.594
-	-	-	-	-	25.74	3.96	4.95	0.495	0.594
central point					26	4	5	0.5	0.6
positive axial points					26.267882	4	5	0.5	0.6
					26	4.267882	5	0.5	0.6
					26	4	5.267882	0.5	0.6
negative axial points					25.732118	4	5	0.5	0.6
					26	3.732118	5	0.5	0.6
					26	4	4.732118	0.5	0.6

Table G.3.: Total sytem costs per replication

Input combinations	Total sytem cost @*l*th replication (@application #2)										Total cost
	1	2	3	4	5	6	7	8	9	10	
i.c. 1	2974.50	3249.28	4351.07	3819.31	3866.04	4034.41	4158.34	4283.42	4847.41	3221.91	3881
i.c. 2	2974.50	3249.27	4351.07	3819.31	3866.04	4138.74	4544.46	3891.75	4034.14	4032.77	3890
i.c. 3	2144.49	4650.53	4077.91	3322.74	4835.23	3642.69	4364.19	3939.71	4093.05	3910.96	3898
i.c. 4	2910.36	3054.69	4079.72	4534.55	3786.16	4205.74	4112.75	3509.12	4548.76	3950.53	3869
i.c. 5	2968.74	3243.52	4345.31	3813.55	3860.28	4028.65	4152.58	4277.66	4841.65	3216.15	3875
i.c. 6	2906.60	3894.78	3922.27	4162.51	4271.60	4207.29	3949.10	4403.57	3117.10	4357.22	3919
i.c. 7	2900.84	3889.02	3916.51	4156.75	4265.84	4201.53	3943.34	4397.81	3111.34	4351.46	3913
i.c. 8	2971.87	3144.12	3979.64	4120.42	4264.11	4462.85	3482.15	3472.38	4631.85	4022.94	3855
i.c. 9	2144.49	4459.83	4009.44	4342.30	4037.79	4189.28	3210.24	5070.82	3895.47	3783.80	3914
i.c. 10	2906.59	4278.05	3894.78	3922.27	4162.51	4271.60	4283.37	3866.27	3879.99	3794.80	3926
i.c. 11	2904.60	3048.93	4073.96	4528.79	3780.40	4199.98	4106.99	3503.36	4543.00	3944.77	3863
i.c. 12	2205.69	4323.85	3968.36	3746.25	4579.18	4010.62	4151.34	4124.97	4284.05	3559.01	3895
i.c. 13	2144.48	4734.73	3726.44	4435.67	3571.43	4242.04	4359.14	4294.95	3785.20	4339.41	3963
i.c. 14	2138.73	4644.77	4072.15	3316.98	4829.47	3636.93	4358.43	3933.95	4087.29	3905.20	3892
i.c. 15	2968.74	3243.51	4345.31	3813.55	3860.28	4132.98	4538.70	3885.99	4028.38	4027.01	3884
i.c. 16	2910.36	3054.69	4079.72	4534.55	3979.38	3264.58	4891.85	3957.32	3817.37	4294.69	3878
i.c. 17	2966.11	3138.36	3973.88	4114.66	4258.35	4457.09	3476.39	3466.62	4626.09	4017.18	3849
i.c. 18	2205.68	4323.85	4067.04	3858.86	4292.39	3851.01	4333.34	3264.27	4739.54	3460.23	3840
i.c. 19	2138.73	4454.07	4003.68	4336.54	4032.03	4183.52	3204.48	5065.06	3889.71	3778.04	3909
i.c. 20	2205.69	4306.03	3907.72	4590.12	3527.44	4338.75	4191.90	4059.08	3646.79	4057.54	3883
i.c. 21	2144.48	4459.83	4009.44	4342.30	3960.66	4127.90	4002.59	3776.49	4214.05	4282.28	3932
i.c. 22	2904.60	3048.93	4073.96	4528.79	3973.62	3258.82	4886.09	3951.56	3811.61	4288.93	3873
i.c. 23	2138.72	4728.97	3720.68	4429.91	3565.67	4236.28	4353.38	4289.19	3779.44	4333.65	3958
i.c. 24	2900.83	3889.02	3916.51	4156.75	4265.84	4277.61	3860.51	3874.23	3789.04	4272.29	3920
i.c. 25	2971.87	3180.87	4052.82	3987.83	4506.67	4432.59	3259.29	4032.02	3870.58	4576.48	3887
i.c. 26	2199.93	4318.09	3962.60	3740.49	4573.42	4004.86	4145.58	4119.21	4278.29	3553.25	3890
i.c. 27	2199.93	4300.27	3901.96	4584.36	3521.68	4332.99	4186.14	4053.32	3641.03	4051.78	3877
i.c. 28	2199.92	4318.09	4061.28	3853.10	4286.63	3845.25	4327.58	3258.51	4733.78	3454.47	3834
i.c. 29	2205.68	4306.03	4403.47	3368.25	4366.27	4415.53	3935.00	3870.40	3853.88	3801.44	3853
i.c. 30	2138.72	4454.07	4003.68	4336.54	3954.90	4122.14	3996.83	3770.73	4208.29	4276.52	3926

i.c. 31	2966.11	3175.11	4047.06	3982.07	4500.91	4426.83	3253.53	4026.26	3864.82	4570.72	3881
i.c. 32	2199.92	4300.27	4397.71	3362.49	4360.51	4409.77	3929.24	3864.64	3848.12	3795.68	3847
centeral point	2817.39	3902.39	3805.63	4225.54	4138.06	4088.09	4288.14	4047.90	4277.02	3839.19	3943
positive axial point 1	2817.39	3902.39	3805.63	4225.54	4138.06	4088.09	4288.14	4047.90	4277.02	3839.19	3943
positive axial point 2	2836.68	3921.68	3824.92	4244.82	4157.35	4107.38	4307.42	4067.19	4296.31	3858.48	3962
positive axial point 3	2835.61	3920.61	3823.85	4243.75	4156.28	4106.31	4306.35	4066.12	4295.24	3857.40	3961
negative axial point 1	2492.14	3703.68	4397.66	4521.43	4493.66	3897.11	4301.80	2818.23	4740.95	4730.45	4010
negative axial point 2	2798.11	3883.11	3786.34	4206.25	4118.78	4068.80	4268.85	4028.61	4257.73	3819.90	3924
negative axial point 3	2867.18	3750.88	4215.77	3256.36	4664.12	3737.27	3814.78	4596.92	3330.91	4740.62	3897

APPENDIX H:

```
%this function performs lack-of-fit F-test
function [F_statistic, F_critical]=lack_of_fit3(beta, alpha, X, costvectorperreplications,
averagecostvector, numberofreplications, totalruns, numberofinputs)

%compute sum of squared residuals and sum of squared errors
averagecostvectorbar=[];
regressionhead=X*beta;
sumsquaredresidual=(costvectorperreplications-
regressionhead)'*(costvectorperreplications-regressionhead);
for i=1:totalruns
   if i<= numberofreplications
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(1)];
   elseif (i>= numberofreplications+1)&&(i<= 2*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(2)];
   elseif (i>= 2*numberofreplications+1)&&(i<= 3*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(3)];
   elseif (i>= 3*numberofreplications+1)&&(i<= 4*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(4)];
   elseif (i>= 4*numberofreplications+1)&&(i<= 5*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(5)];
   elseif (i>= 5*numberofreplications+1)&&(i<= 6*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(6)];
   elseif (i>= 6*numberofreplications+1)&&(i<= 7*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(7)];
   elseif (i>= 7*numberofreplications+1)&&(i<= 8*numberofreplications)
      averagecostvectorbar=[averagecostvectorbar;averagecostvector(8)];
   elseif (i>= 8*numberofreplications+1)&&(i<= 9*numberofreplications)
```

```
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(9)];
elseif (i>= 9*numberofreplications+1)&&(i<= 10*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(10)];
elseif (i>= 10*numberofreplications+1)&&(i<= 11*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(11)];
elseif (i>= 11*numberofreplications+1)&&(i<= 12*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(12)];
elseif (i>= 12*numberofreplications+1)&&(i<= 13*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(13)];
elseif (i>= 13*numberofreplications+1)&&(i<= 14*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(14)];
elseif (i>= 14*numberofreplications+1)&&(i<= 15*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(15)];
elseif (i>= 15*numberofreplications+1)&&(i<= 16*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(16)];
elseif (i>= 16*numberofreplications+1)&&(i<= 17*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(17)];
elseif (i>= 17*numberofreplications+1)&&(i<= 18*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(18)];
elseif (i>= 18*numberofreplications+1)&&(i<= 19*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(19)];
elseif (i>= 19*numberofreplications+1)&&(i<= 20*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(20)];
elseif (i>= 20*numberofreplications+1)&&(i<= 21*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(21)];
elseif (i>= 21*numberofreplications+1)&&(i<= 22*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(22)];
elseif (i>= 22*numberofreplications+1)&&(i<= 23*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(23)];
elseif (i>= 23*numberofreplications+1)&&(i<= 24*numberofreplications)
    averagecostvectorbar=[averagecostvectorbar;averagecostvector(24)];
elseif (i>= 24*numberofreplications+1)&&(i<= 25*numberofreplications)
```

```
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(25)];
    elseif (i>= 25*numberofreplications+1)&&(i<= 26*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(26)];
    elseif (i>= 26*numberofreplications+1)&&(i<= 27*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(27)];
    elseif (i>= 27*numberofreplications+1)&&(i<= 28*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(28)];
    elseif (i>= 28*numberofreplications+1)&&(i<= 29*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(29)];
    elseif (i>= 29*numberofreplications+1)&&(i<= 30*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(30)];
    elseif (i>= 30*numberofreplications+1)&&(i<= 31*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(31)];
    elseif (i>= 31*numberofreplications+1)&&(i<= 32*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(32)];
    elseif (i>= 32*numberofreplications+1)&&(i<= 33*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(33)];
    elseif (i>= 33*numberofreplications+1)&&(i<= 34*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(34)];
    elseif (i>= 34*numberofreplications+1)&&(i<= 35*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(35)];
    elseif (i>= 35*numberofreplications+1)&&(i<= 36*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(36)];
    elseif (i>= 36*numberofreplications+1)&&(i<= 37*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(37)];
    elseif (i>= 37*numberofreplications+1)&&(i<= 38*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(38)];
    elseif (i>= 38*numberofreplications+1)&&(i<= 39*numberofreplications)
        averagecostvectorbar=[averagecostvectorbar;averagecostvector(39)];
    end
end
```

```
sumsquaredpurerror=(costvectorperreplications-
averagecostvectorbar)'*(costvectorperreplications-averagecostvectorbar);
sumsquaredlackfit=sumsquaredresidual-sumsquaredpurerror;

%perform F-test
dof1=numberofinputs-size(beta,1);
dof2=totalruns-numberofinputs;
F_statistic=(sumsquaredlackfit/dof1)/(sumsquaredpurerror/dof2);
F_critical = finv(1-alpha,dof1,dof2);
```

APPENDIX I:

Matlab program codes for Appendix I can be found in the attached cd document with the name “robust_rsm_3.m”.

Objective function @ *fmincon*

```
%this function creates the objective function for the minimization
function obj = objective3(x)
obj = (-2.00829e+006) + 43093.4*x(1) + 313036*x(2) + 41741.4*x(3) + (1.48e+006)*x(4) + (1.18377e+006)*x(5) - 6009.64*x(1)*x(2) - 5114.92*x(1)*x(3) -...
31250.1*x(2)*x(3) - 52091.2*x(4)*x(1) - 312501*x(4)*x(2) + 222195*x(4)*x(3) - 40164.2*x(5)*x(1) - 260418*x(5)*x(2) +...
178239*x(5)*x(3) + 169.08*x(1)^2 - 6.47815*x(2)^2 - 200.209*x(3)^2;
```

Minimization function

```
[x,fval,exitflag]=fmincon(@objective3,[26; 4; 5; 0.5; 0.6],[0 1 1 0 0], 15, [], [], 26;0;0;0.495;0.594], [50;15;15;0.505;0.606])
```

BIOGRAPHICAL SKETCH:

Seda Eyigün was born in in Balıkesir, on April 3, 1981.

She completed the high school in Balıkesir Lisesi, in 1999.

She received his B. Sc. degree in Computer Engineering from Galatasaray University, İstanbul, in 2005. Since 2005, she is in the M. Sc. program in Industrial Engineering of Galatasaray University.

For the completion of the program she has studied on "Robust Optimization through Response Surface Methodology". She wrote her first academic article in 2008 with Yrd. Doç. Dr. M. Ebru Angün.