THE DEVELOPMENT OF AN AUTOMATED DISCRETE-EVENT SIMULATION OPTIMIZATION SYSTEM

By

ZEYNEP AYSEGUL KARACAL

Bachelor of Science
Middle East Technical University
Ankara, Turkey
1982

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1988

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May,1995

THE DEVELOPMENT OF AN AUTOMATED

DISCRETE-EVENT SIMULATION

OPTIMIZATION SYSTEM

Thesis Approved:

Allen C. Schaumann

Thesis Adviser

M. P. Lerskill

David B. Pratt

Manjunath Kamath

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my gratitude to the faculty and staff of the School of Industrial Engineering and Management for making my stay at Oklahoma State University a rewarding and enjoyable learning experience.

I would like to express my appreciation to my major advisor, Dr. Allen C. Schuermann, for his guidance, support and assistance in the development of this study and for his editorial guidance in preparing this manuscript. He was always willing to be of assistance for any problem concerning this study and other difficulties encountered during this time.

I extend my thanks to my committee members, Dr. M. Palmer Terrell, Dr. Manjunath Kamath, Dr. David B. Pratt and Dr. William Warde for their suggestions, assistance and cooperation. I also extend my thanks to Dr. Joe H. Mize, who initially served on my committee, for his suggestions and assistance.

To my friend, Dr. Camille Deyong, I offer my heartfelt thanks for not only being my friend but also for her constant support and encouragement.

I would like to acknowledge the support provided by members of my family here and in Turkey. My parents, Guner and Orhan Izgu, left their home and stayed with me during this study. There are not any words in any language to express my appreciation to them. I want to thank them for their love, support, understanding and for their faith

and encouragement to achieve my goals. Without their support, this work would not have been finished.

My love and deepest appreciation is extended to my husband, Dr. Cem Karacal, for his constant support, encouragement and understanding. Last, but not least, thanks to my children, Sîla Gizem and Ada Irem, for their love, patience, and affection. I wish to dedicate this dissertation to my daughters who made everything worthwhile.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Simulation is a problem solving procedure for defining and analyzing a model of a system. Simulation is the use of a computer model to mimic the behavior of a complicated system to gain insight about the system performance. Simulation has often been used as a means of explaining and analyzing the behavior of an existing large complex system or predicting the performance of a new system. One should keep in mind that a simulation model does not explicitly describe the relations, but mimics how it operates under given conditions. Therefore simulation does not provide optimal solutions to problems.

As a system becomes more complex and the number of important factors increase, it is not always practical or possible to express relationships through a set of mathematical equations, and the decision maker is forced to use other analytical tools to solve the problem. In cases where mathematical modelling is infeasible, simulation becomes an alternate way to solve difficult problems. Although simulation is not an optimization procedure, a simulation model is often used to measure the performance of a system which is to be optimized. The purpose of simulation optimization is to optimize the objective function and identify the settings of the design parameters which will result in this optimum level. The objective function in a simulation model is expressed in

terms of outputs from the simulation model. Therefore a simulation model can be viewed as a stochastic objective function in which inputs are converted into a value for the objective function by executing the simulation model. For large systems, even identifying an optimum for a single output variable can become a real challenge. There is a need to develop a consistent and structured approach to eliminate this deficiency of a simulation-based approach. This study addresses this issue by using output analysis techniques and simulation optimization procedures.

The major steps in discrete-event simulation process (see Pritsker(1986)) are:

1. Problem Formulation
2. Model Building
3. Data Acquisition
4. Model Translation
5. Verification
6. Validation
7. Strategic and Tactical Planning
8. Experimentation
9. Analysis of Output
10. Implementation and Documentation

Based on the assumption that the first eight steps of the process are completed successfully, this study addresses only the output analysis step.

Due to the stochastic nature of the input and output parameters, it is important to analyze the results using appropriate output analysis techniques. Otherwise, without any statistical analysis, the results cannot be used with any degree of reliability. Output

analysis techniques can be divided into two major categories, techniques for terminating systems and techniques for non-terminating systems. These techniques, with respect to the type of the simulation, are discussed in Chapter II. Output analysis techniques for steady state simulations are considered in this study.

Problem Statement

As the size and complexity of the real world systems increase, simulation becomes a popular decision making tool. Computer simulation may be viewed as a "black box" in which various input parameters produce various responses. Figure 1 shows the black box view of a computer simulation.

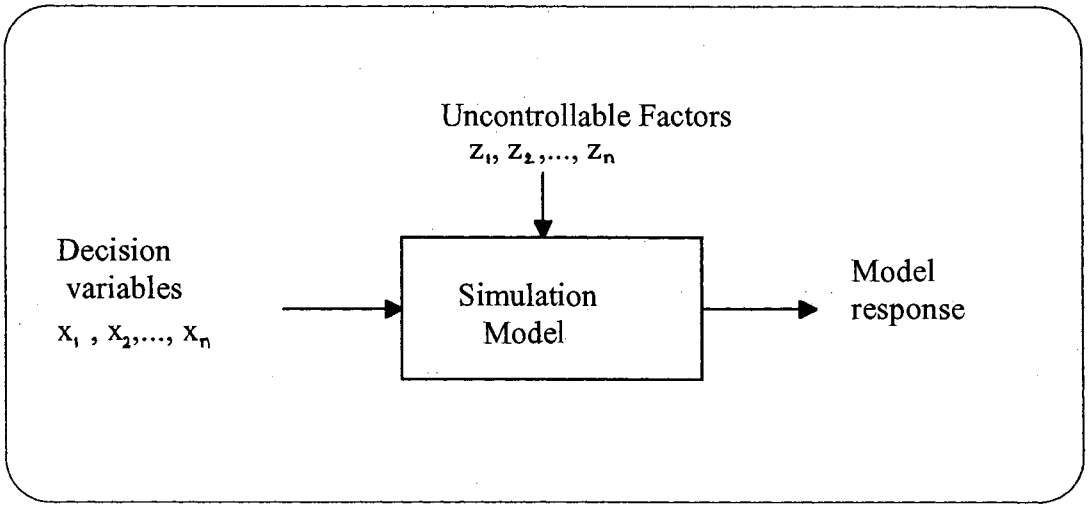


Figure 1. Black Box View of Computer Simulation

The general optimization problem can be formulated as follow:

$$\text{Min (Max)} \quad f(x_1, x_2, \ldots, x_n)$$

In some cases, the objective of the simulation is to obtain the best solution for the given system. In that sense, that problem is similar to an optimization problem to be solved by mathematical techniques. Briefly stated, a simulation optimization problem is an optimization problem where the objective function, constraints or both are responses that can only be evaluated by computer simulation. These functions are often stochastic in nature as well.

The major differences between a simulation optimization problem and an ordinary optimization problem are as follow:

1. The functional form of the response is not known.

   $$\eta = g(x_1, \ldots, x_n)$$

   There is no objective function explicitly stated. The objective function is expressed in terms of output data generated by the simulation. Although the response is a function of the input variables, the mathematical relation which forms this function is not known.

2. The system responses contain random error (ε) due to the unpredictable effects of the uncontrollable factors.

   $$y = g(x_1, \ldots, x_n) + \varepsilon$$

Thus, the simulation model can be viewed as a stochastic objective function. The responses generated by the simulation model are estimates of the expected values of the

true responses. Therefore, the classical notion of an optimum is inapplicable due to the presence of random error.

The common approach to find a good solution, which is close to the desired optimum, is to run the simulation model with some inputs, which are believed to lead to a good solution, analyze the results, compare the alternative solutions and then select the "best" possible solution. This process usually degenerates into a trial and error process involving an inordinate amount of human effort and computer time.

Although the literature is full of various optimization techniques and their isolated applications in simulation, there are very few approaches that combine simulation with search procedures and incorporate the statistical aspects. Bengu and Haddock (1986) combined a simulation generator for an inventory system and an optimization routine consisting a set of direct search procedures. Bengu (1987) also used simulation-optimization approach by combining direct search procedures and statistical tools. Bengu developed a program containing derivative-free search algorithms is appended to a deterministic simulation model to optimize multivariable, single objective functions. This study will address simulation optimization problem. This need is beyond the mere collection of procedures to form a software package. The characteristics of the problem and user's needs must be taken into consideration by creating an intelligent environment which will satisfy the user's goal automatically and take the burden of computations and frequent decision making away from the user.

Combining simulation with search procedures reduces this effort considerably and makes simulation an efficient decision making tool. The search for the best response of the simulation experiment involves two interrelated problems:

1. Obtaining the desired statistical precision and accuracy of the simulation results via output analysis.
2. Investigation of the best response for the system (simulation optimization).

The first problem requires the elimination of the initial bias and the estimation of the variability of correlated time series data. The second problem requires the comparison of alternatives and an investigation of the best response by using search procedures.

Although the computation time for the search algorithms is negligible when compared to the time necessary to evaluate the objective function, the overall response time remains critical in real world situations. Therefore, an effort must be made to reduce response time which may require reducing the number of simulation runs, an initial screening of a factor, or the interaction of the analyst.

Combining a simulation model with an optimization algorithm can be a problem due to the interface requirements between them. The optimization algorithm must serve as an executive to the simulation program and determine the initial conditions for each run, the number of runs and length of each run. Since the simulation model must be invoked by the optimization program, it is difficult to use existing optimization packages. For example, the use of FORTRAN based software may prevent the use of other than FORTRAN based simulation languages. The language compatibility between simulation models and optimization can be expanded by the emergence of new simulation languages that allow interfaces with several different programming languages. Also, the use of commercially available software packages may create interface problems between the modules of the proposed automated simulation optimization system.

Briefly stated, the problem statement for this research can be summarized as:

> To develop an automated integrated discrete event simulation system which can be used as a framework for modelling, optimizing and analyzing, via simulation, multivariable single response unconstrained problems.

Organization of the Dissertation

This research is described in detail in the following five chapters. Chapter II reviews current simulation software, simulation output analysis techniques and simulation optimization techniques. Output analysis techniques with respect to the type of the simulation are discussed in this chapter. Optimization techniques are classified according to the type of the search method they employ, and are covered in terms of technical discussion, applications, advantages and disadvantages in Chapter II. Chapter III presents the research goal, objectives and scope of the research. Chapter IV discusses the research methodology used in this research effort and gives a detailed description of each method used in this study. Chapter V contains the demonstration and analysis of the system developed and the interpretation of the results obtained. Finally, the research effort is summarized, the contributions of the research are listed and the recommendations for further research are presented in Chapter VI.

CHAPTER II

LITERATURE SURVEY

Simulation Software

In recent years, there has been a tremendous increase in the number of simulation software packages available on a wide variety of computers. Developments in computer technology, reduced computing costs, rapid changes in the manufacturing technology, availability of graphical animation, and increased interest in simulation has led to the development of many simulation software packages. When the number of software packages was small and the features offered were alike, the selection of the right software was not difficult. But with the increasing number of software available and the vast variety of features offered, the decision-making process may become a long and tedious one.

The selection of the right software is a two-step process. The first step is to develop an understanding of the software. One should know how the software are classified and determine the necessary features. The second step is the selection of the software through a systematic search. Haider and Banks (1986) classified simulation software into three levels: system, application, and structural. Classification of the software at the system level is based on the type of the system. There are two basic

types of systems: discrete-event and continuous. In discrete-event systems, the state variables change only at discrete points in time. Most of the manufacturing systems are modeled by discrete event simulations. In continuous simulation, variables change continuously over time. Continuous simulation models consist of sets of differential, difference equations. They may also contain stochastic components. This study will focus on discrete event simulations.

At the application level, simulation software can be classified as special purpose and general purpose. Special purpose simulation software or simulators are designed to model specific environments. General purpose products or simulation languages can be used for virtually any system.

At the structural level, classification is based on the modelling orientations. Since simulators are mostly data-driven, this classification is for simulation languages. The first modelling orientation is event scheduling in which a system is modeled by defining the changes that occur at event times. The second orientation is activity scanning in which the modeler needs to identify the conditions to start and end each activity. The last orientation is process orientation which provides a description of the flow of entities through a process. The process is defined either by user written routines or a network representation of blocks/nodes. Figure 2 shows the classification of the simulation software at these three levels.

Table I (adapted from Banks(1991)) shows four classes of simulation modelling tools. The first class consists of spreadsheets (e.g. LOTUS). They model random events using an @RAND or similar command. They are limited to small systems and

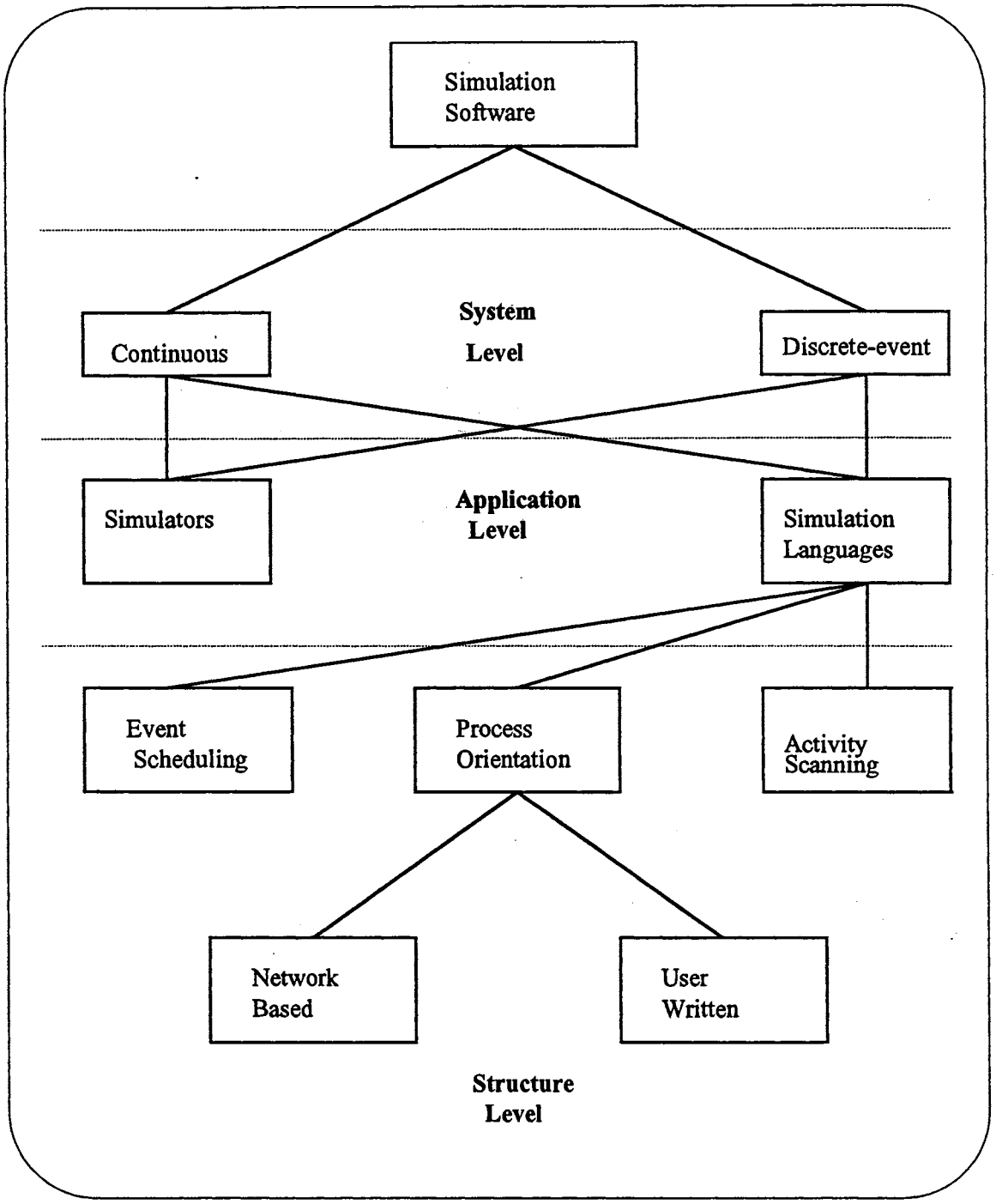


Figure 2. Classification of Simulation Software

Consideration	Spreadsheets	Rapid Modelling Tools	Simulators	Simulation Languages
Development Time	Minimal	Minimal	Moderate	Moderate to Substantial
Model Control and System Complexity	Poor for Dynamic Systems	Vague / Moderate	Must Comply with Software Constraints	Excellent/ Virtually Any Systems
Output	User Developed and Defined	Statistically Adequate, Report Oriented	Report Oriented Varies by Package	User Defined Reports
Accuracy/ Fidelity	Generally Inaccurate for Dynamic Systems	Good for Rough Cut Planning	Varies with level of Assumptions	Excellent
Training	Minimal	Moderate	Moderate	Moderate to Substantial
Environment Best Suited to	Static Systems Deterministic Operations	Low Complexity Probabilistic Operations/ Multiple Alternatives	Medium Complexity/ Probabilistic Operations/ Data Available/ Specific Applications	High Complexity/ Probabilistic Operations/ Data Available/ General Applications

Table I. Classification of Simulation Modelling Tools (adapted from Banks (1991))

the outputs are usually in the form of graphs. Seila and Banks (1990) showed the application of spreadsheets for simulation modelling. The second category contains rapid modelling tools like MPX (formerly MANUPLAN) (see Banks (1991)). These tools can give an idea about system performance without going into great detail.

The third category is the simulators which are data driven simulations. They usually require no programming. However the robust ones either allow programming within the simulation or use an interface with another language or compiled code. Simulators remove the need for language syntax and they have a good user interface. The ability of a simulator to model a system varies from software to software. The analyst has to make some assumptions and evaluate the model within the capabilities of the simulator. Banks et al. (1991) examined four manufacturing simulators currently available (SIMFACTORY II.5, XCELL+, WITNESS and PROMODELPC) and tried to solve two sample manufacturing problems using these four simulators. They also listed the desirable features of the simulators.

The last category contains simulation languages. Up to the 1960's, simulation was performed by using programming languages such as FORTRAN. This started to change about 1961 when GPSS was introduced by Geoffrey Gordon . Early versions of SIMSCRIPT by Markowitz, Hausner and Karr, and GASP by Kiviat and Cooker were introduced in 1963 and 1964 respectively (see Banks & Carson (1984)). About 1970, the second generation of these languages was introduced. GPSS/H was introduced in 1977 and Pritsker and Pegden combined GASP and Q-GERT to form SLAM. SIMAN was introduced in 1983. At that time, the number of simulation languages increased dramatically. Current simulation languages allow very detailed programming of almost

any realistic problem. They offer customized output along with their standardized output.

Since spreadsheets and rapid modelling tools are not suitable for detailed simulation analysis and simulators are limited to medium complexity problems and good for specific applications, only simulation languages will be considered in this study.

The desirable features of simulation software can be classified into five groups. They are: input, processing, output, environment, and cost features. Table II summarizes these features.

Input features: One of the most important features is modelling flexibility. Since no two systems are the same, the language should have the capability of modelling different systems. Portability is another feature to be considered. The programs written for one computer should be able to be run on other types of computers. The syntax should also be easy to understand and consistent. Another input feature is interactive debugging which allows the user to control the accuracy and speed of the modelling process. Input data analysis capability, which defines the distribution of input parameters, and input flexibility are also desirable features for any simulation language.

Processing features: Model size and execution speed are especially important if the models are run on PCs. Execution speed is also crucial for the analysis of large and complex systems which may require numerous runs. Most of the modern manufacturing systems have material handling systems which may include Automated Guided Vehicles (AGVs), transporters (e.g. forklift trucks), Automated Storage/Retrieval Systems (AS/RS), cranes and robots. These systems are often difficult to model. Therefore, the availability of material handling modules may reduce the

FEATURES		
INPUT	* Interface to other software * Input data analysis capacity * Input flexibility * Interactive debugger * Modelling flexibility	* Portability * Syntax * Modelling conciseness
PROCESSING	* Execution speed * Model size * Reset * Random Variate Generator * Attributes * Independent Replications * Global Variables	
OUTPUT	* Customized reports * Standardized reports * Graphics * File creation * Tracing * Data base management * Confidence intervals	
ENVIRONMENT	* Ease of use * Documentation * Animation - ease of development - quality of picture - CAD interface * Customer support - training - technical support - updates	* Ease of learning * On-line help - portability - smooth movements

Table II. Desirable Features for Simulation Software

modelling time. Since the systems to be modeled may have random features (uncontrollable inputs) , it is important that the software have good statistical capabilities. A simulation package should contain a wide variety of random variate generators including basic distributions such as uniform, exponential, gamma, etc. The software should also have the ability of making independent replications and to specify the warm-up period .

Output features: It is desirable for a simulation package to have customized reports along with time saving standardized reports. Graphical displays (bar charts, pie charts, histograms) are also desirable. Data base management is another helpful features which organizes the outputs from various runs for future references.

Environment features: The software should be easy to learn and use. The detailed documentation of the language should accompany the software. Animation has become a desirable feature for the simulation packages. It is a useful tool for program debugging, testing new strategies, and communication between manufacturing personnel and managers. Animation features should include easy development, CAD interface, a quality picture and smooth movement of the icons on the screen. Customer support is another important issue. Some users may require on-going support through training and technical support.

Cost Features: Cost of the software can range from several hundred dollars to several thousand dollars. One should also consider hardware requirements for the specific software as an additional cost.

Although programming languages such as FORTRAN, or C can be used for simulation, they do not directly provide any facilities to make the user's job easy.

The user has to program all of the details of the simulation such as event scheduling, time advance algorithms, statistics collection, and random number and variate generation. For small models, these programming languages can be used as a learning tool, but for large systems, to use these languages can be quite complex and cumbersome.

On the other hand, high level simulation programming languages such as SLAM, SIMAN, SIMSCRIPT, etc., are specially designed for model building (see Law & Kelton (1991)). They provide the user with a choice of orientation (process or event) and built-in facilities (statistics collection, random variate generators, animation, graphics and report generators). Therefore this study will only focus on high level simulation languages. Table III displays basic information on some of the popular simulation languages.

GPSS/PC, SIMAN, SIMSCRIPT II.5, SLAM II and SLAMSYSTEM were chosen to demonstrate and compare basic features. All of these simulation languages are general purpose and they can combine both discrete and continuous simulations, can start the simulation at empty state and can save the model state at the end of each run for later restarts. They all have animation capabilities. SLAM II has this capability through TESS for the mainframes, SIMAN has animation through CINEMA and SIMSCRIPT has animation through SIMANIMATION for its PC version.

As it can be observed from Table III, there is very little difference between the languages listed in the table. For this study, SLAMSYSTEM was chosen as the simulation language. FORTRAN was chosen as the programming language for the search the procedures and the output analysis module.

LANGUAGE

FEATURES	GPSS/PC	SIMAN	SIMSCRIPT II.5	SLAM II	SLAM-SYSTEM
Event / Process Scheduling	P	E, P	E, P	E, P	E, P
Interactive Debugger	Yes	Yes	Yes	Yes	Yes
Available on Which Computer Types	PC	PC, Mini, Main,WS[1]	PC, Mini, Main, WS	PC,Mini, Main,WS	PC
Programming Language Accessible		FORT, C		FORT	FORT
Animation	Yes	Yes Cinema	Yes	Yes TESS	Yes
Material Handling	No	Yes	No	Yes	No
Standard reports	Yes	Yes	Yes	Yes	Yes
Customized reports	Yes	Yes	Yes	Yes	Yes
Graphics	Yes	Yes	Yes	Yes	Yes

[1] WS: Work Station

Table III. Basic Features of Simulation Languages

Simulation Output Analysis

Simulations are used to understand the behavior of the system under study. Frequently, the system's behavior is summarized by the values of one or more measures, such as mean waiting time or mean service time. Unfortunately in many studies, a large amount of money and time is spent on the development of the model and programs, but little effort is made to analyze the simulation output data. A common practice is to make inferences about the system from a single simulation run of arbitrary length without appropriate statistical analysis. This practice may occur because simulation is often viewed simply as an exercise in computer programming. In fact, it is a computer based statistical sampling experiment. Therefore appropriate statistical techniques must be used.

Another reason for the lack of output analysis is that the simulation outputs are nonstationary and autocorrelated. Therefore classical statistical analysis based on independently identically distributed (iid) observations are not directly applicable.

As a result of the inadequacy of classical statistics, extensive research has been done to develop appropriate procedures for the output analysis. In the following sections, the types of simulation with regard to the analysis of the output and the techniques used to analyze both types of simulation will be covered.

Types of Simulation with respect to Output Analysis

Simulations can be classified into two categories: terminating and nonterminating.

Terminating Simulations:

A terminating simulation is one that runs for some duration of time T_E where E is a specified event which stops the simulation. Event E is specified before the simulation begins and depends on the nature of the simulation and the purpose of the analysis. The measure of performance for terminating the simulation explicitly depends on the initial state of the system. Therefore initial conditions must be carefully chosen (The following examples are adopted from Law and Kelton(1982a), and Banks and Carson(1984)).

Example 1:

A bank opens at 9:00 am with no customers present (initial conditions at time=0) and closes at 5:00 pm (physically terminating at $T_E = 480$ minutes). In this case one might want to study the interaction between customers and tellers over the entire day.

Example 2:

A company sells a single product and wants to determine how much to order to minimize the average monthly cost of operating the inventory system for the next 10 years ($T_E = 10$ years).

Some systems, such as Example 1, start each day in the same state , operate for a specified period and then terminate. On the other hand, some systems can operate indefinitely. Yet someone may still be interested in the system's behavior between certain periods or up to the time when the "X"th item is produced.

The choice of the initial states is very important. If the intent was to study the bank in Example 1 from 11:00 am to 2:00 pm, it would not be correct to start up the system with no customers. Either real system data should be used for this period or the simulation should run from 9:00 am without collecting any statistics prior to 11:00 am.

A terminating simulation is appropriate if the system shuts down regularly or the system has a natural duration time or the short term responses of a new system under some conditions are needed to be studied.

Nonterminating Simulations:

A steady state or nonterminating simulation is a simulation for which the desired measures of performance are defined as limits as the run time of the simulation goes to infinity. When the system reaches steady state , the initial conditions no longer affect the system's behavior. In another way, the behavior of observations does not depend on when they are collected. As an example, in a manufacturing system, the behavior of the system will be the same after some period of time regardless of the amount of the initial inventory.

Although the type of the simulation appropriate for the system might be obvious for some systems, in some cases either type of simulation might be appropriate. In that case, the analyst must choose a type depending on what is to be learned about the system.

Statistical Analysis Techniques for the Analysis of Simulation Output

The output data from a simulation model presents random variability due to the use of random numbers to produce the input data. If the performance of the system is measured by a parameter θ, the result of a set of simulation experiments will be an estimate $\hat{\theta}$ of θ. The variance of $\hat{\theta}$ is used to measure the accuracy of $\hat{\theta}$. The usual approach to determine the accuracy of the estimator is to construct a Confidence Interval (CI) for the true measure. The methods used to estimate the variance depend on the type of the system that is being simulated. The next section classifies the statistical analysis procedures into two classes for terminating and nonterminating simulations.

Terminating System Procedures

As mentioned before, the length of the simulation is defined by the system conditions and the estimated parameters explicitly depend on the system. Therefore, the output must be generated by independently replicating simulation runs using the same initial conditions. The independence of runs is accomplished by using different random numbers for each replication. The observations will be independent and identically distributed (iid) and classical procedures for independent data can be used. In that case, the estimator of the parameter X_j from the jth replication can be considered an iid

random variable (rv) with finite population mean and variance. The estimate of mean $\mu = E(X)$ from n replications will be

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \qquad (1)$$

and the sample variance s^2 (unbiased estimator of σ^2) will be

$$s^2 = \frac{\sum_{j=1}^{n} (X_j - \bar{X})^2}{n-1} \qquad (2)$$

Under the assumption that the X_j's are normal random variables, the 100(1-α)% confidence interval for the true value of μ is given by

$$\bar{X} \pm t_{n-1,\alpha/2} \frac{s}{\sqrt{n}} \qquad (3)$$

It must be noted that the CI will be approximate because the X_j will rarely be normally distributed. However the violation of normality can be ignored for large n based on the Central Limit Theorem. Therefore, the violation of normality will not affect the results of equation 3.

This procedure is called a fixed sample size procedure ($n \geq 2$) (see Law and Kelton (1991,1982a)). One disadvantage of this procedure is that the user has no control over the CI half length. But in many simulation studies, procedures were developed to construct a CI with a small absolute precision (actual CI half length) or a small relative precision (ratio of the CI half length to the magnitude of the estimator) . Law and Kelton

(1982a) present a sequential process which adds new replications one at a time until a CI with a specified precision is constructed. If the precision of the CI is not crucial, one may choose to use a fixed-sample-size procedure. Also one must consider the cost of each replication. Since the observations X_j are themselves averages, the assumption of normality is reasonable based on the Central Limit Theorem. As the number of replications increases, the standard error gets smaller. Although a precise CI is desirable, it may not be affordable. Law and Kelton (1991) recommend at least 3 replications to assess the variability of X_j.

Nonterminating System Procedures

Nonterminating simulations produce output data that are independent of the initial state of the system. The simulator must stop the simulation after "n" observations or a specified length of time T_E is reached. The sample size n or T_E is a design choice which has nothing to do with the nature of the problem. The simulation length should be chosen with several considerations in mind:

1. The bias in the point estimators due to the initial conditions.
2. The desired accuracy of the point estimators.
3. Budget constraints.

When analyzing a steady state simulation, the analyst must deal with two major problems: initialization bias and autocorrelation.

Initialization bias which is caused by using unrealistic initial conditions can lead to wrong inferences about the system, especially when several independent runs are used to construct the CI. Startup policies, for setting initial conditions or truncation procedures for specifying the truncation point at which data can be considered for estimations, can be used to minimize initialization bias.

Pritsker (1986) suggests three basic rules for setting the initial state of the system.

1. Start the system empty and idle. Although it is easy to implement this rule, it may not be a good representation of the system. This is hardly the case for manufacturing and information systems.
2. Start the system at steady state mode. Probably it is the best way to start up the system, but it may be very difficult to determine the steady state mode for large systems.
3. Start the system at the steady state mean. For this rule, either a pilot study or an analytical analysis of the system must be run.

Kelton (1989) and Wilson and Pritsker (1978a) list procedures to minimize the initial state of the system so that the initial bias will be minimized.

There are several methods to eliminate initial bias. One method is to collect data on the system and use the data to specify initial conditions. This approach, if the system exists, may require a lot of data collection. Otherwise it is impossible to implement. In spite of this difficulty, it is always better to use available data on the system rather than making unrealistic assumptions about the system such as starting the system empty or idle.

Another start up procedure is to divide each simulation run into two phases: the initial transient phase and the data collection phase. The problem in this approach is to determine the truncation point. Unfortunately there are no proven techniques to determine how much data to delete to minimize initialization bias. The common approach is to make a pilot run and select a time based on that run. One can try to minimize the initial bias by selecting the appropriate initial conditions, running the simulation long enough to make the initial bias insignificant and/or dividing the simulation into an initialization phase and a data collection phase.

Each approach presents problems in terms of implementation. There is no widely accepted procedure to reduce initialization bias. Pritsker (1986) lists a limited summary of some truncation rules. The survey by Wilson & Pritsker (1978a) contains many techniques for controlling initialization bias. Law & Kelton (1982a) have developed a procedure based on independent replications, deletion of data and time series regression techniques. Schruben (1982) presents a test for detecting initialization bias using a hypothesis testing framework.

Steady state simulations produce data which are independent of the initial state of the system. Performance measures are defined in terms of the steady state behavior of the system. A great amount of effort has been done to develop point and interval estimators for steady state simulations. If $X_1, X_2, \ldots, X_n$ are the waiting times in a system then the steady state mean waiting time is

$$\nu = \lim_{n \to \infty} \frac{\sum_{i=1}^{n} X_i}{n}$$

It is assumed that ν is independent of the state of the system at time 0. There are two general approaches in the literature to construct the estimators for steady state simulation.

1. Fixed-Sample-Size Procedures. A simulation run or several independent runs are performed and one of the available techniques is used to construct the CI.
2. Sequential Procedures. The idea is to increase the length of the simulation run sequentially until an acceptable CI can be constructed.

Fixed Sample Size Procedures: Law (1983) categorizes fixed sample size procedures into four basic types.

1. Those that seek independent observations.
2. Those that seek to estimate dependence among the output variables.
3. Those that exploit a special structure of the underlying process.
4. Those based on standardized time series.

REPLICATION:

This approach is also used for terminating simulations and falls into the first category. If the initialization bias can be reduced to a negligible level, this method can be used to estimate point estimators and to construct confidence intervals. Otherwise the resulting confidence intervals might be misleading. The number of replications has no effect on the bias which is affected only by deleting more data or increasing the simulation length. Therefore, if the initial bias exists, increasing the number of replications may produce short coverage of the CI around the wrong point.

Let X_j be the sample mean of the last m-r observations in the jth replication where r is the warm-up period. Then X_j's become iid and can be substituted into equations (1), (2) and (3) for point estimates and a 100(1-α)% CI.

There are several potential difficulties with this method. First, to choose the number of observations to be deleted is very critical. If too few are deleted, every replication will contain some transient observations which will make every sample mean X_j biased. On the other hand, if too many observations are deleted, the CI will be wider than necessary. Secondly, this method uses data inefficiently. Thirdly, each run starts with the same initial conditions which are not representative of the steady state. Finally, one must interrupt the simulation to collect the data and reinitialize the runs for each replication.

The major advantage of this method is that the sample means are guaranteed to be iid. One may use this method if he/she is sure that the system reaches steady state quickly and run lengths can be limited.

BATCH MEANS METHOD:

This method attempts to take independent observations from the process, but only a single simulation run of length (n) is made. Data from the stationary portion of the run are divided into (m) batches of (k) consecutive observations each (n=mk). Let Y_j be the sample mean of k observations in the jth batch. If the batch size k is chosen large enough, then the Y_j's will be approximately uncorrelated and normally distributed. If this is the case, the batch means can be treated as iid random variables and the usual statistical methods can be used to construct a CI for the mean.

Due to potential sources of error associated with the correlation between batch means, procedures which seek to find an appropriate size for k are needed. Gross and Harris (1974) suggested an approach to fix the number of batches (m) and increase the batch size (k) until the estimated correlation between adjacent batch means is less than a specified tolerance. Fishman (1978a) also proposed a method for fixed m by choosing k based on the Van Neumann ratio. Fishman concluded that his method works well for large sample sizes if the process is not too positively correlated. Law (1977) conducted a comparison of the replication method and batch means method. He showed that the actual coverage of a desired 90% CI is 27% by using the replication method and 72% by using the batch means method for a M/M/1 queue with $\rho = .9$ where m=20 and k=64 . He repeated the same experiment with 10 batches instead of 20 and showed that the actual coverage of a 90% CI was 62% and 76.7% for the replication method and the batch means method respectively. As a result of this empirical study, Law reported that the violation of the normality assumption has no major impact on both methods as long as the batch size (k) is larger than 20.

Other papers that discuss the batch means method are Conway (1963), Mechanic & McKay (1966), Schmeiser and Kang (1981), Schmeiser (1982), Schreiber & Andrews (1979) and Adam (1983).

The advantage of this method is that it uses data more efficiently. The data are deleted only once . Therefore, unlike the replication method, misjudgments about the length of the initial transient period will have little effect on the batch means. Law and

Kelton (1984) concluded that the batch means method is generally superior to the other methods that have been proposed in terms of producing the most accurate CI's.

AUTOREGRESSIVE METHOD (Parametric Modelling):

This method does not attempt to obtain independent r.v.'s from the data. It employs estimates of the correlation structure to obtain an estimate of the variance of the sample mean. It was developed by Fishman (1971, 1973, 1978b) and assumes the process is covariance stationary and can be represented by a pth order auotoregressive model (AR(p)). Law and Kelton (1984) tested this method of parametric modelling and found that the actual coverage of the CI may be less than the desired coverage if the sample size is too small.

Andrews & Schreiber (1982) generalize Fishman's model by assuming that the process can be represented by a autoregressive-moving average model ARMA(p,q). They used a technique developed by Gray, Kelley & McIntire (1978) to determine the order of p and q. But empirical results from two queueing systems showed that the coverage of the CI is less than the desired coverage and inconsistent with increasing sample size. According to Pritsker (1986), " In our experience, parametric modelling of the time series obtained from a simulation model has not produced reliable estimates of the variance of the sample mean."

SPECTRUM ANALYSIS:

This method also employs estimates of the correlation structure to determine an estimate of the variance of the sample mean (Fishman (1978b), Law (1983)).

Given the lag h covariance, R_h, where

$(\Sigma |R_h|^2 \angle \infty)$ the spectral density function is defined as

$$g(\alpha) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} R_h e^{-i\lambda h} \qquad -\pi \le \lambda \le \pi$$

and

$$m = \sum_{h=-\infty}^{\infty} R_h \approx 2\pi g(0)$$

The estimation of $\text{var}(\bar{Y}) = \frac{\hat{m}}{n}$

where n is the number of observations and $\hat{m}$ is the estimate of the spectral density function at zero frequency. This technique is complicated and requires a sophisticated statistical background on the part of the analyst. Estimating the spectral density function requires the determination of the number of covariance weighting functions to apply to the estimated autocovariance obtained from finite observations.

REGENERATION METHOD:

This method identifies random times at which the process probabilistically starts over and uses these regeneration points to obtain independent random variables. When a system regenerates itself, its future behavior is independent of the past. The pattern of development after each regeneration point is the same. For example in a M/M/1 queueing system, regeneration points are the times when a customer arrives and finds the system empty. Arrivals and departures of the customers will be independent of the arrivals and departures of the past customers and will be a replica of those after empty state because of the iid nature of service and interarrival times. For the output process $(Y_i, i \ge 1)$, assume that there is a sequence of random times $1 \le B_1 \le B_2 \le \ldots$

called regeneration points at which the process starts over with the same probabilistic structure. The portion of the process between two consecutive regeneration points is called the regeneration cycle (Y_i, $B_j \leq i \leq B_{j+1}$). Define

$$N_j = B_{j+1} - B_j \qquad j=1,2,...$$

and

$$E(N_j) < \infty .$$

$$Z_j = \sum_{i=B_j}^{B_{j+1}-1} Y_i \quad .$$

The random vectors $U_j = (Z_j , N_j)^T$ are iid. Then the steady state average response is given by

$$\nu = \frac{E(Z)}{E(N)}$$

This method was developed simultaneously by Fishman (1973) and by Crane and Iglehart (1975). Law and Carson (1979) also developed an alternative regenerative approach known as the Jacknife approach.

STANDARDIZED TIME SERIES (STS) METHOD:

This method uses the methodology of weak convergence of functions of stochastic processes (see Billingsley(1968), Goldsman and Schruben(1982), Schruben(1983), Schruben et al. (1983)).

The standardized time series model assumes that the process (Y_i, $i \geq 1$) is

strictly stationary and phi-mixing (the process is phi-mixing if Y_i and Y_{i+j} become independent as j becomes large).

This method divides n observations into m batches of length k (n = mk). $Y_j(k)$ is the sample mean of the jth batch where Y(n) is the grand sample mean and the point estimator of ν. If n is large , the grand sample Y(n) will be approximately normally distributed.

$$Y(n) \sim N(\nu, \frac{\sigma^2}{n})$$

where

$$\sigma^2 = \lim_{n\to\infty} n(\mathrm{Var}\bar{Y}(n))$$

The statistic A, which is asymptotically independent of Y(n), is defined as

$$A = \left[\frac{12}{k^3 - k}\right] \sum_{j=1}^{m} \left\{ \sum_{i=1}^{k} \left[\bar{Y}_j(k) - \bar{Y}_{i+(j-1)}(k) \right] \right\}$$

It can be shown that (Schruben(1983))

$$A \sim \sigma^2 \chi^2_m$$

$$\frac{(\bar{Y}(n) - \nu) / \sqrt{\frac{\sigma^2}{n}}}{\sqrt{\frac{A}{m\sigma^2}}} = \frac{\bar{Y}(n) - \mu}{\sqrt{\frac{A}{mn}}}$$

For large k;

$$\left([\bar{Y}(n) - \nu]\ \sqrt{\frac{\sigma^2}{n}}\ \sqrt{\frac{(A/\sigma^2)}{m}} = (Y(n) - \nu)\ \sqrt{\frac{A}{mn}} \sim t(m)\right.$$

Therefore an approximate 100(1-α)% CI for ν is given by

$$Y(n) \pm t_{m,1-\frac{\alpha}{2}} \sqrt{\frac{A}{mn}}$$

The advantage of this method is its computational simplicity. Schruben (1983) has shown that this method works well in terms of coverage for large k's. But there is a need to investigate the performance of the method for small values of k.

SEQUENTIAL PROCEDURES

Sequential procedures sequentially determine the length of the simulation run needed to construct an acceptable CI. As observed in the fixed length procedures, all of the methods display the problem of insufficient coverage of the CI. Also in addition to the coverage problem, the simulator may want to have some control over the CI procedures in terms of the absolute precision (β) or relative precision (γ). In case of fixed length simulations, there is no way to know in advance the magnitudes of γ and β.

The procedures developed so far fall into two categories : either the process is assumed to be regenerative or the process is assumed to be nonregenerative.

Fishman's (1977) and Lavenberg and Sauer's (1977) procedures fall into first category. Fishman's procedure attempts to construct a CI for ν with absolute precision β. The output data are grouped into blocks of n' adjacent regeneration cycles. A point estimator h_j for ν from the jth block is formed. The random variable h_j's are iid. The simulation is continued until enough h_j's are collected so that $\beta < \gamma$ for the CI produced.

The procedure also checks the bias and normality of the h_j's. If they are unbiased and normal, the final CI is formed as the average of the h_j's. Otherwise the block size n' must be increased and the process must be repeated.

Fishman's process seeks to obtain iid random variables in order to use classical statistical methods directly. This process has the disadvantage of being based on regenerative methods which might limit the application of the method in real world simulations. The selection of the minimum width γ which is not explained in the procedure might require a prior run. Also this method shows small coverage if β is not chosen small enough.

Lavenberg and Sauer (1977) form a CI for ν such that the ratio of its half length to its midpoint does not exceed a fixed constant $\gamma > 0$. Their stopping rule is based solely on the relative width criterion and backed up by an asymptotic theory. According to the empirical results in Law & Kelton (1982b), Fishman's procedure performs better.

Mechanic & McKay's (1966) procedure is based on the batch means method. In this procedure, N observations are broken into batches and an average autocorrelation estimate is computed from each batch mean. The batch size is successively increased until the sequence of autocorrelation estimates satisfies certain conditions and these batch means are used to construct the CI. If the conditions are not met, the number of observations N is increased and the steps are repeated until a suitable batch size is obtained. This procedure does not have a built-in mechanism to force the CI to become small.

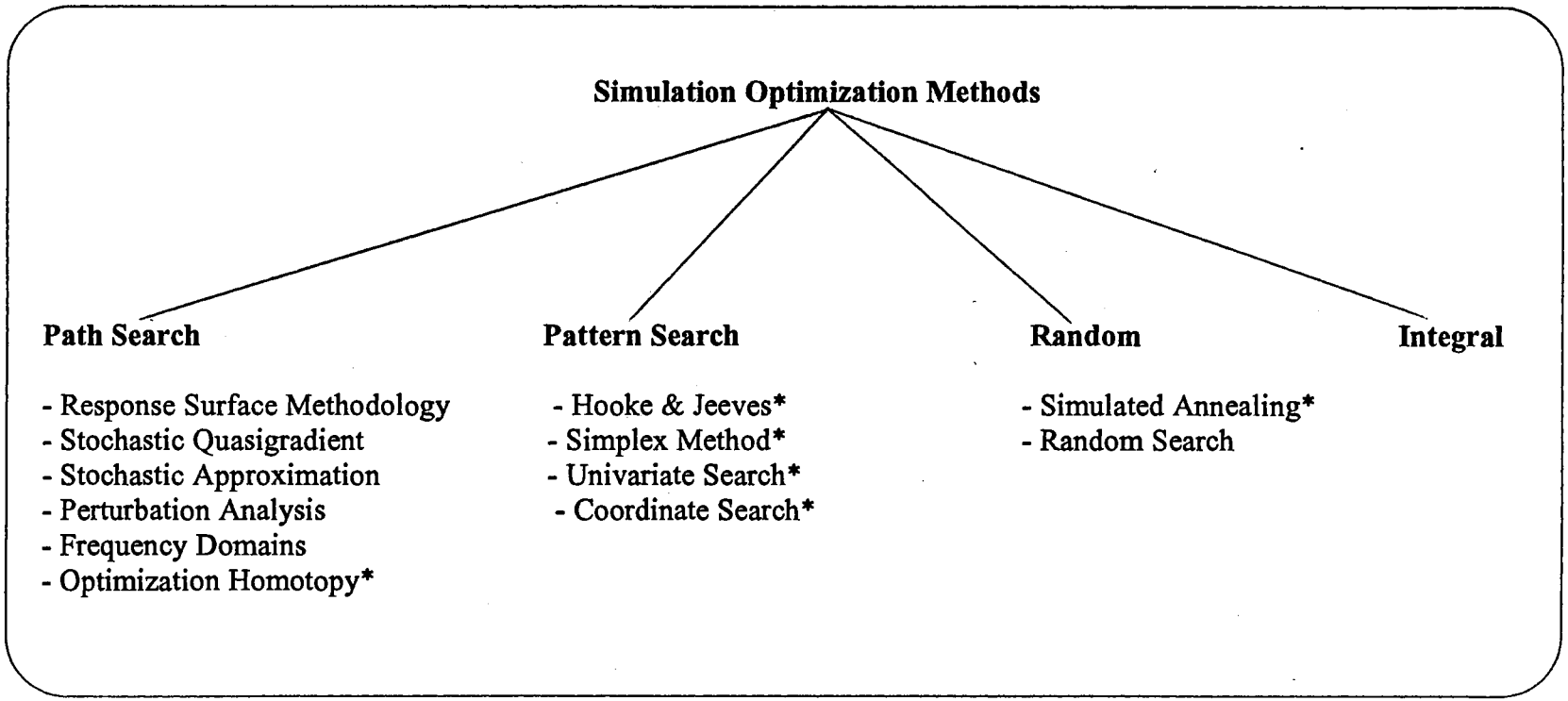
Law & Carson's (1979) method is also based on batch means. It divides m observations from a single run into 400 batches of size k ($m = 400k$). If the estimated lag 1 correlation between the 400 batch means is less than a threshold value $c = 0.4$, the same m observations are divided into 40 batches and these 40 batch means are considered to be uncorrelated and used to construct a CI for ν using the batch means approach. If the estimated lag 1 correlation is not less than c or if the actual relative precision is not less than γ, then m is increased and above steps are repeated. Law & Carson's method appears to be the most preferable in nonregenerative applications.

The advantage of the sequential method is that if the technical assumptions of the method apply, the precision of CI is guaranteed. On the other hand, the computer time required for the simulation is not predictable in advance. Another difficultly with these methods is that they must be built into the simulation. Other than these difficulties, sequential methods are the most preferable approach to computing CI's.

Simulation Optimization

Simulation is often considered to be an attractive alternative for analyzing complex systems for which the analytical solution is very difficult or impossible. Although simulation is not an optimization procedure, a simulation model is often used to measure system performance which is to be optimized. The intent of simulation optimization is to optimize the objective function and identify the settings of the design parameters. This desire usually initiates the trial and error use of simulation which is performed until

parameter settings are found that satisfy the desired goal. This process requires many simulation runs. Since the objective function cannot be expressed as an explicit function of the design parameters, the objective function is expressed in terms of outputs from the simulation experiment. It has been shown that simulation can be integrated with analytical models and/or optimization schemes to accelerate this trial and error process to find the "best" parameter setting which results in the "optimum" level (see Starr (1966)). Moore and Lee (1989) presented a method which optimizes a closed loop factory line balancing in a semiconductor wafer fabrication facility. Farrel et al (1975) surveyed early works in simulation optimization. Other articles in the literature searches include Smith (1973b), Rustagi (1981), Birta (1984), Meketon (1987), Jacobson & Schruben (1989) and Safizadeh (1990). Figure 3 shows the classification of simulation optimization methods. These methods can be classified into four categories according to the search method they employ: path search methods, pattern search methods, random search methods and integral methods. As shown in Figure 3, the methods are either developed specially for the systems or adopted from nonlinear optimization techniques by replacing the objective function value in the algorithm with the estimate of the function obtained from the simulation run. The first section covers some of the optimization techniques and their applications in simulation optimization. The last section covers multiresponse optimization approaches and their applications.



* Methods adopted from Nonlinear Optimization Techniques

Figure 3. Classification of Simulation Optimization Methods

Optimization Techniques and their Applications

Response Surface Methodology (RSM) : RSM is a set of techniques used to design a set of experiments that will provide adequate and reliable measurements of the response, determine the mathematical model that best fits the data and determine the optimal setting of the experimental factors which produce the desired objective . The relationship between the response and the input variables is given by the response function

$$\eta = \Phi (x_1, x_2, ..., x_n)$$

where η is the true response and Φ is the response function. In complex situations the exact form of η will be unknown and is approximated by a polynomial equation. Denoting the approximate value of η by y, the optimization effort is focused on the estimate of the expected value of y. A second order polynomial equation for k variables can be written as

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \beta_{ij} x_i x_j + e$$

or

$$y = \sum_{i=0}^{k-1} \sum_{j=i}^{k} \beta_{ij} x_i x_j + e \qquad \text{where } x_0 = 1$$

Most of the common models used to approximate the polynomial are of degree one and two. Designs used to fit first order polynomials are called first order designs . Some first order designs are 2^k factorial designs, simplex designs, 2^{k-p} fractional factorials and

Plackett-Burman designs (see Plackett and Burman(1946)). In general, the initial simulation experiment is based on a first order design to determine the path of steepest descent (ascent). Additional simulation runs are then made until no improvement in response is observed. Then a new first order design is run around the "optimal point". This is called Phase I of RSM. Phase I is repeated until there is a lack of fit in the first order design. A second order design is then chosen and a second order polynomial is fit to the response. This is called Phase II of RSM. Calculus is used to determine the stationary point of this polynomial. A detailed list of first and second order designs is presented in Myers (1971) and Safizadeh (1990). Related works can be found in Smith and Mauro (1982). They studied screening factors that would reduce the number of variables in the polynomial.

Daughtery & Turnquist (1981) used RSM with constraints based on the cost to run the simulation. Heller and Staats (1973) used RSM to solve problems subject to costs and constraints by modifying Zoutendijk's method of feasible directions. Another use of RSM with simplex experimental designs is presented in Mihram (1970). Cooley and Houck (1982) looked at variance reduction strategies for RSM simulation. Nozari, Arnold and Pegden (1987) and Tew and Wilson (1987) have developed statistical strategies that enhance the applicability of the Schruben & Margolin (1978) variance reduction design in metamodel estimation for a simulation response. Biles (1977) and Rees et al. (1985) studied multiple response fitting and multiple response optimization. Montgomery and Bettencourt (1977) used the Geoffrion & Dyer interactive vector minimal algorithm for multi-response optimization. King and Fisher (1989) developed a prototype system (BARBS- Bottleneck Analysis Rule Based System) to identify

manufacturing bottlenecks. They combined RSM, simulation and expert analysis heuristics for bottleneck identification. Safizadeh and Thornton (1984) reviewed RSM in optimization of simulation experiments. Early reviews can be found in Farrell et al. (1975), Farrell (1977), Brightman (1978), Montgomery (1979) and Montgomery and Evans (1975).

The advantage of RSM is that it is easy to understand, is based on statistical theory and is easy to implement. The major disadvantage of the method is that it requires a large number of runs. For example, when k factors are being investigated, at least k+1 points (runs) are needed to calculate the path of steepest descent before any search is actually done. This disadvantage can be overcome by screening variables and using important ones. Cochran and Chang (1990) used two-stage group screening methodology to identify important variables to be used in RSM to find the optimal solution. Group screening is based on the aggregation principle and requires prior knowledge about the variables. After grouping variables into several groups, each group is treated as a single factor. Then a factorial design is used to identify important factors and reduce the number of computer runs. After the identification of the major variables, RSM can be applied. Cochran and Chang used this approach along with RSM for a flight simulation. Another disadvantage of RSM is that the actual implementation might be difficult because of the need for a higher order polynomial to fit the surface. Also it should always be kept in mind that whenever RSM is used, a statistical model is analyzed. Therefore, the analyzer must be careful with the assumptions in the simulation and statistical modelling stages.

As quoted from Safizadeh and Thornton (1984), " RS designs in conjunction with gradient based optimization techniques and search methods appear to best satisfy the optimization objective of simulation."

Frequency Domain (FD) Methods: One of the recent strategies for optimization is the frequency domain method. This method was originally introduced for screening the factors in the simulation runs that have significant effect on the output. In a frequency domain experiment, input variables are changed during a run according to the sinusoidal or rectangular oscillations. If the simulation response is sensitive to changes in a particular factor then the response can be predicted by oscillating that factor. The main assumption in FD experiments is that the output response can be modeled as a polynomial function of the input levels. Such a function is called a meta-model or simulation response surface regression model. Simulation response surface models are high level mathematical relationships which are helpful in understanding the complex relationship between inputs and outputs.

Two basic tasks in simulation response surface modelling are the identification of the functional form of the response surface model and the estimation of the coefficients of the response surface model. Schruben and Cogliano (1987) presented an experimental method for identifying an appropriate model for a simulation response surface.

For frequency domain experiments to be applicable in identifying a model for a given system, the system must have :

- Parameter settings that can be changed during an experimental trial;
- A system response that can be observed at periodical intervals;

- A response that can be adequately modeled as a time-variant linear combination of products of powers of the functions.

A FD experiment requires at least two runs: a control run and a signal run. In the control run, input factors are held constant at their nominal values. A control run identifies natural cycles in the response. In a signal run, the input factors are changed according to the sinusoidal oscillations. The frequency assigned to a particular factor is called its driving frequency. For example, the sinusoidal wave for a particular input factor can be represented as

$$\Phi_i(t) = \alpha_i \cos(2\pi\omega_i t + \delta)$$

where α_i is the amplitude , δ is the phase shift, t is the time index and ω_i is the driving frequency of the ith variable . The response spectrum has a peak ω_i at corresponding α_i . The contribution of each frequency to the variability of the time series is measured by a function called the spectrum. As linear system theory states, a sinusoidal input produces a sinusoidal output at the same frequency. The output spectrum $f_g(\omega)$ can be written in terms of the input spectrum $f_\Phi(\omega)$ as follows

$$f_g(\omega) = G^2(\omega)\, f_{\Phi_i}(\omega) + f_e(\omega)$$

where G is the gain function which describes how the system amplifies or attenuates oscillations at different functions and $f_e(\omega)$ is the spectrum for random disturbance. If e is white noise then $f_e(\omega)$ is constant for all ω_i .

In a multiple factor linear system, the response spectrum and input spectrum are related by

$$f_g(\omega) = \sum_{i=1}^{k} G_i^2(\omega)\, f_{\Phi_i}(\omega) + f_e(\omega)$$

The three steps for designing FD experiments are (See Schruben and Cogliano (1987)):

1. The selection of a set of driving frequencies for input factors;
2. The determination of amplitudes;
3. The assignment of driving frequencies to each input variable.

Schruben (1986) and Schruben and Cogliano (1987) presented the steps for metamodel identification. Jacobson, Russ, and Schruben (1991) presented a heuristic algorithm for the selection of driving frequencies which maximize the minimum space between the term indicator frequencies. Mitra and Park (1991) introduced a technique for performing FDEs which uses the simulation clock as the oscillation index and they demonstrated this technique for a network of queues.

According to the empirical studies done, the number of runs required to identify the important factors by the frequency domain method are much less than conventional run-oriented simulation experiments where each setting of input values requires a separate run. Although the frequency domain approach is not by itself an optimization technique, it can be used to identify the factors which can be estimated by other optimization techniques. For example, it can be used with classical RSM to indicate when Phase II of RSM should be started.

The frequency domain approach has several advantages. First, several input factors can be studied in the same run. Second, nonlinear effects can be detected with no additional experimentation. Third, high order terms in the response surface can also be identified without additional runs. Along with these advantages, FD also has some limitations too. First, the user should have knowledge about FD and spectral analysis. Second, the implementation of changes inside the model can be a very difficult task. Third, selection of the experimental region is critical. Although the larger the region, the more power there will be in detecting input factor effects, it may put the simulation in an unstable region for too much of the time.

Perturbation Analysis (PA) : A new methodology PA has been used in the optimization of queueing networks. PA was first developed by Y.C. Ho. PA is an analytical technique that calculates the sensitivity of performance measures of a discrete event dynamic system (DEDS) with respect to the system parameters by analyzing its sample path. General overviews on PA are presented in Ho (1985), Ho (1987), Suri (1989).

Four types of PA and related works are listed below.

1. Infinitesimal PA (IPA). See Cao (1985,1988), Heidelberger et al. (1988), Ho and Cao (1983), Suri (1987), Suri and Zazanis (1985), Zazanis and Suri (1986).
2. Extended Perturbation Analysis (EPA) for systems that can be represented as a continuous time Markov chain. (see Cao (1987), Ho and Cassandras (1983), Ho and Cao (1983), Ho et al. (1979)).

3. Smoothed Perturbation Analysis (SPA) . This method is an extension of IPA based on conditional probability (see Gong (1988), Gong and Ho (1987)).

4. Finite Perturbation Analysis (FPA). This method introduces perturbations and propagates them while observing the nominal path and limits the calculations by extrapolating to predict the effects of such changes. (see Cao (1987), Ho & Cassandras (1983), Ho, Eyler and Chien (1979), Ho, Eyler and Chien (1983).

PA tracks and records certain statistics during the simulation such as the sensitivities of the parameters with respect to the simulation response. It uses the chain rule from calculus to estimate the gradient using only one simulation run. For example, using a M/M/1 queue example adopted from Jacobson & Schruben (1989), let ω be the simulation response (e.g. average waiting time), s be the measure of interest (e.g. service time of a customer) and x be the factor of interest (e.g. service rate), then according to the chain rule

$$\frac{\partial \omega}{\partial x} = \frac{\partial \omega}{\partial s}\frac{\partial s}{\partial x}$$

where $\frac{\partial s}{\partial x}$ is assumed to be known and evaluated from the distribution function F(s,x). The primary assumption of IPA is that the order of events for the original path and the perturbed path stay the same for both x and $x + \nabla x$ where ∇x is an infinitesimal perturbation. In other words, IPA assumes that the perturbation of a parameter is small enough not to change the order of events in the simulation run.

The concept of PA can be used in simulation studies. It is usually easy to implement the PA algorithm in simple simulation models. But for large and complex simulation models, the implementation of PA becomes less straightforward. Also, the implementation of PA is limited to a certain class of queueing networks. For example the assumptions of PA hold true for GI/G/1 queues, single class Jackson Networks and Tandem Network with blocking. Related studies can be found in Cao & Ho (1983), Ho and Cassandras (1983), Ho and Eyler and Chien (1979,1983).

Ho (1987), Kumar (1984), Suri (1983a, 1983b, 1987), Suri & Zazanis (1988) reported the theoretical justification of PA and other related issues to PA in their papers. Heidelberger et al. (1988) discussed the limitations and potential weakness of PA and explained why PA can give very poor gradient estimates in certain simulation models because of the assumption of the same order of events in the existence of a perturbed factor.

PA provides the gradient information which could be used with optimization procedures. The application of PA can be seen in the sensitivity analysis of discrete event simulations (see Suri & Dille (1985)), single run optimization (Suri & Leung (1987,1989)) and on line control/improvements of DEDS (Cassandras (1987)).

Random Search: Random search uses a random approach to choose the parameter settings. Before it can proceed, the upper and lower limits for each parameter must be defined. There are two basic approaches to conducting random search. The first method involves a random sampling of points from the grid given by the factorial design. In the other approach, the number of runs are specified a priori and the "best response" is chosen as optimum when the number of runs is exhausted.

The major advantage of random search is that there is no limit on the number of runs and its simplicity. However , this search does not cover the search region thoroughly, there is no guarantee for a global optimum and the method does not use the information from previous runs. Smith (1973a) presented the possibility of using random search. Garcia-Diaz et al. (1983) used the random search method along with the Out-of-Kilter algorithm for the analysis of a production distribution system. Fox (1984) used the idea of random search with quasi random numbers to minimize the discrepancy of the samples. The idea of using quasi random numbers in conjunction with random methods is a promising future research field.

Simulated Annealing: Simulated annealing is an iterative stochastic search method which has been designed for deterministic multivariate combinatorial problems. It is analogous to the physical annealing process whereby material is gradually cooled so that the minimum energy state is reached.

The simulated annealing concept was introduced by Metropolis et al. (1953) who developed a simple algorithm to provide an efficient simulation of atoms in equilibrium at a given temperature. Kirkpatrick et al.(1983) and Cherny (1985) applied this concept to deterministic optimization problems. Vecchi & Kirkpatrick (1983) applied simulated annealing to global wire routing. Cherny presented a Monte Carlo algorithm to find approximate solutions for the travelling salesman problem.

Since its introduction, SA has been applied to solve all kinds of optimization problems arising in computer science, engineering and image recognition. Recently, Bulgak and Sanders(1988) have demonstrated one application of the SA algorithm with a

simulation to optimize buffer sizes in automatic assembly systems. Manz et al.(1989) showed the possibility of using SA to optimize an automated manufacturing system. Hajek (1988) established cooling schedules for optimal annealing. Wilhelm and Ward (1987) used the SA method to solve a quadratic assignment problem. Lee and Iawata (1991) proposed an annealing algorithm to solve a part ordering/release problem in FMSs.

SA randomly generates moves, and checks whether the cost of the new configuration is acceptable based on a parameter T, sometimes called "temperature ". If the cost decreases, the move is accepted. Otherwise the move is accepted with a probability which is a function of T and the increase in the cost (e.g. $\frac{1}{1+e^{\frac{(c_i - c_{i+1})}{T}}}$ where c_i , c_{i+1} are the costs).

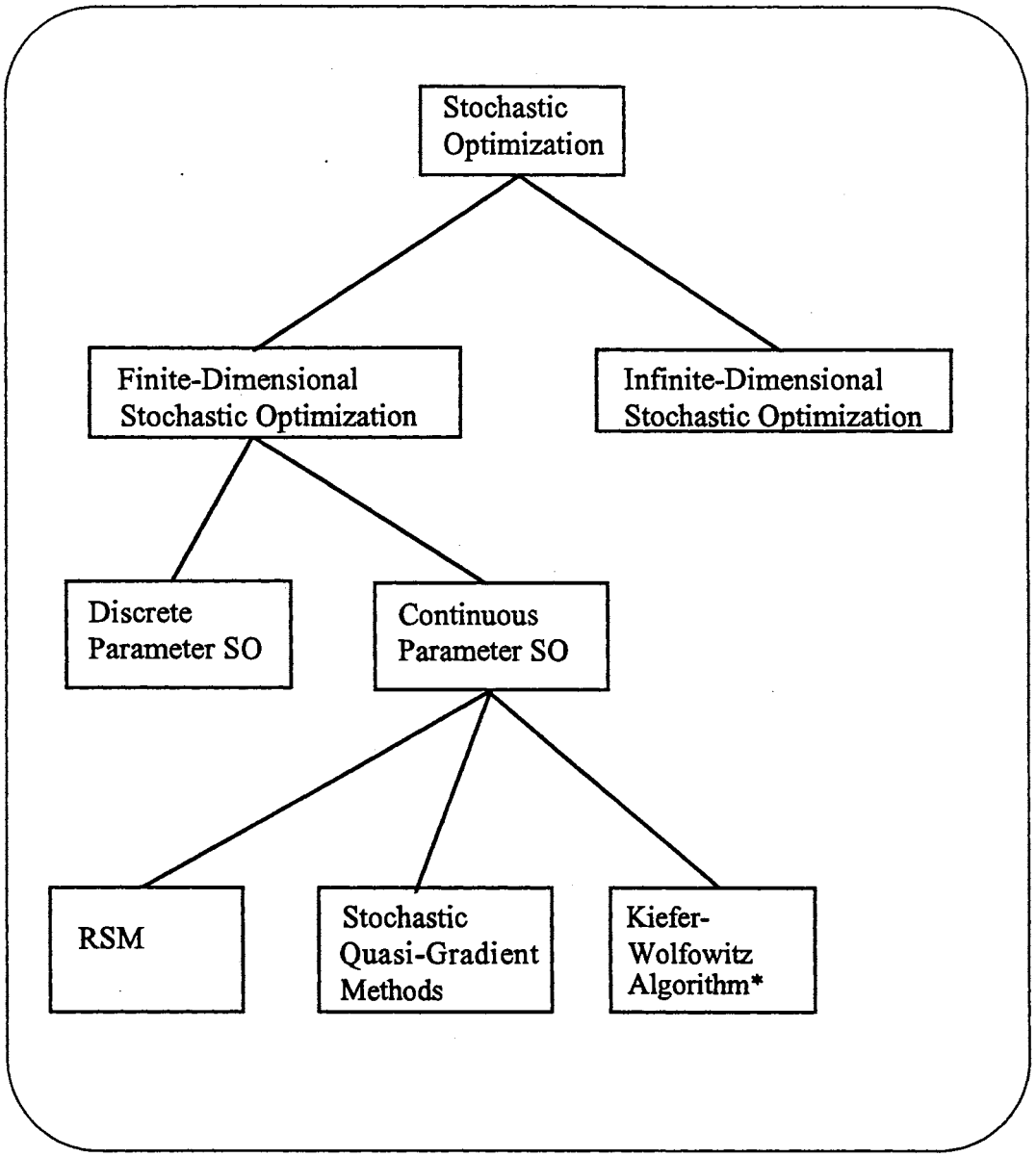
The algorithm behavior is strongly dependent on the existing temperature. At high temperatures, the probability of a hill-climbing move is higher. When T is zero, no hill-climbing move is accepted. For each value of T, a certain number of moves is generated, accepted or discarded before T is changed. This sequence of nonincreasing temperatures defines an annealing schedule which is determined by the set of parameters (e.g. starting temperature, number of iterations, rate of decrease of T and stopping criteria)

The selection of an annealing schedule plays an important role in the efficiency of SA. If the temperature decreases too quickly, even local optimum might be missed. If the temperature decreases too slowly, an excessive number of function evaluations may be required. Also the number of iterations to be performed at each temperature also

affects the performance of the algorithm. Through demonstrations, the choice of an initial solution does not affect the success of the algorithm (Hajek(1988)). The effect of the characteristics of the combinatorial optimization problem, such as the behavior of the cost function on the behavior of the simulation algorithm, are yet to be discovered. The question of which problems are suitable for SA has been addressed lately and is a promising field for future research.

The primary advantage of the algorithm is its ability to deal with a large number of problems naturally and effectively. Also it is relatively easy to implement SA algorithms to solve new problems. On the other hand, SA has limited efficiency when it is compared with the heuristics that are specially developed for certain problems. SA algorithms start at high temperatures and very slowly lower the temperature which results in long computation times. The efficient annealing algorithms stated in the literature are mostly problem dependent because of the structure of the cost functions. When the cost function is too irregular or flat, it becomes difficult to reach the global optimum.

Stochastic Optimization: Stochastic optimization is concerned with the general problem of optimization under uncertainty. Glynn (1986) presented a survey of the algorithms for stochastic optimization. Figure 4 shows a classification of stochastic optimization methods. Finite stochastic optimization can be divided into two categories: discrete parameter stochastic optimization and continuous parameter stochastic optimization. Discrete parameter methods are not developed as well as the continuos case and are under investigation. On the other hand, continuous parameter methods are more robust and can be applied to a greater variety of problems. Continuous algorithms



* Based on Robbins-Monro Algorithm

Figure 4. Classification of Stochastic Optimization Methods

can be divided into three classes: response surface methodology, stochastic quasi-gradient algorithms and the Kiefer-Wolfowitz (K-W) algorithm based on the Robbins-Monro (R-M) algorithm. Since response surface methodology was covered previously, the rest of the continuous parameter algorithms will be discussed.

The stochastic quasi-gradient (SQG) method is a stochastic approximation method for solving general constrained optimization problems with nondifferentiable, nonconvex functions. The SQG method was developed by Ermoliev and Shor (1968). SQG methods use finite-difference Monte Carlo estimates for gradients which will be used in gradient-based deterministic optimization algorithms. A survey of these methods and their applications can be found in the paper by Ermoliev(1983). SQG methods can be applied to many fields including optimization of stochastic systems, identification and reliability of a system , inventory control and manufacturing systems. Liu and Sanders (1988) presented an application of the SQG method to the performance optimization of asynchronous flexible assembly systems (AFAS).

Kiefer and Wolfowitz (1952) proposed a stochastic approximation algorithm (K-W) based on the Robbins-Monro (R-M) algorithm using finite difference approximations to the gradient. The Robbins-Monro algorithm is not an optimization procedure, but is a root finding algorithm. It can be used to find the root of the gradient of the objective function in optimization applications. The details of the algorithm can be found in Robbins and Monro (1951). The difficulty of this approach is to estimate an unbiased gradient when the objective function values are found from simulation. Azadivar and Talavage (1980) developed an algorithm to optimize a stochastic system by

using the principles of the stochastic approximation method. Ruppert et al. (1984) applied SA to a Monte Carlo simulation of a fish harvesting model.

Single Run Optimization (SRO) : Single run optimization is a kind of stochastic optimization method which optimizes the simulation model in a single run by saving computational effort and computer time. This is achieved by estimating the gradient of the objective function, then updating the parameters based on the estimated gradient while the simulation is running. Single run optimization was originally suggested by Meketon (1987). Suri and Zazanis (1988) have done a preliminary study of single run optimization. As mentioned before, Perturbation Analysis can be used to estimate the gradient of the objective function. Since this estimate is available during the simulation run, it is possible to change the parameter values during the run and get an estimate of the optimum at the end of the simulation run. This is the basic idea behind the single run optimization. Suri and Leung (1989) developed two single run optimization methods: the Perturbation-Analysis- Robbins-Monro-Single-Run (PARMS) algorithm and the Kiefer-Wolfowitz-Single-Run (KWSR) algorithm. The basic difference between these algorithms is the way that the gradient of the objective function is estimated. PARMS uses PA for the estimates of the gradient, while KWSR uses a finite-difference procedure.

Since both algorithms change the parameter values continuously during the simulation, the process never reaches steady state and remains in a transient state. Therefore, neither the theoretical results of PA nor the stochastic approximation procedures (R-M and K-W) hold. PARMS and KWSR were implemented to optimize

an M/M/1 queue with respect to the mean service time . According to the test results, PARMS showed better performance in terms of run length and average percentage errors.

Suri and Leung (1987) applied a single run optimization algorithm to optimize cycle time of a closed loop flexible assembly system by implementing the single run optimization algorithm and estimation of the gradient via perturbation analysis in the SIMAN language. Leung and Suri also investigated the finite time behavior of the RM algorithm and the single run optimization algorithm. They showed that the single run optimization algorithm converges faster than the RM algorithm.

Although the single run optimization algorithm gives exciting and promising results, stopping criteria, iteration length and convergence of the algorithm need further investigation. SRO algorithms based on PA require some additional computations to be done during a simulation.

Pattern Search Methods: These methods do not require gradient estimates. In general, they set a pattern and then move in the direction of the pattern to obtain a new point which leads to a better solution. The most common techniques are Hooke and Jeeves method, coordinate search, Nelder and Mead Simplex method and rotating coordinates. The differences between the techniques are the local explorations and the method of computing step sizes. The details of the algorithms can be found in Bazaara and Shetty (1979). These methods replace the objective function value with an estimate obtained from simulation. Since these algorithms are designed for deterministic cases, the optimum solution is not guaranteed in the presence of random error.

A coordinate search changes one parameter at a time in the simulation runs. The rest of the parameters are kept the same while the chosen parameter is increased until no improvement in the objective function is observed. The algorithm terminates when no change is observed during a pass or the number of runs is exhausted.

The Hooke and Jeeves method employs two types of moves: exploratory and pattern moves. The exploratory move serves to establish a direction of improvement and the pattern move projects the solution vector to a new point in the solution space to restart the exploratory move. If the objective function continues to improve , the length of the pattern move (step size) is increased. Otherwise, the search retracts and the length of the pattern move is decreased. Pegden and Gately (1980) applied this method to a decision-optimization module for SLAM. Bengu and Haddock (1986) also applied this method to problems in SIMAN.

Another well-known technique is the Nelder and Mead Simplex Method. In this method, simulation runs are performed at the vertices of the initial simplex. The point resulting from the worst objective function value is replaced by a new point found via reflection through the centroid of the simplex. Depending on the value of the objective function, the simplex is either expanded, contracted or remains the same .

Integral Methods: Evtushenko (1971) introduced the integral optimization approach with an algorithm for Lipschitz continuous functions. His algorithm is based on space covering and is specifically designed for global optimization. So far this algorithm has not been successfully implemented due to the difficulty of obtaining a Lipschitz constant in a stochastic environment. Zheng (1986) developed an integral

optimization approach without using the Lipschitz constant. The algorithm requires considerable work at each iteration and converges slowly.

Multiresponse Optimization

Computer simulation can be considered as a black-box which combines values of n decision variables to produce values for a set of m response variables η_j. The relation between the input variables and the system response can be defined by the response function f_j.

$$\eta_j = f_j(x_1,x_2,...,x_n) \qquad j=1,2,...,m$$

The responses are also affected by some uncontrollable factors. This is depicted in Figure 5 (adapted from Biles and Swain (1982)).

The purpose of the computer simulation is to evaluate the various policies and parameter values for operating a system and to find the optimum values for the decision variables. Determining the values of the decision values x_i's, which may include an infinite number of possible values, requires the employment of classical optimization. But one must consider two important characteristics of simulation models:

1. The response functions are not usually known,

$$\eta_j = f_j(x_1,x_2,...,x_n) \qquad j=1,2,...,m$$

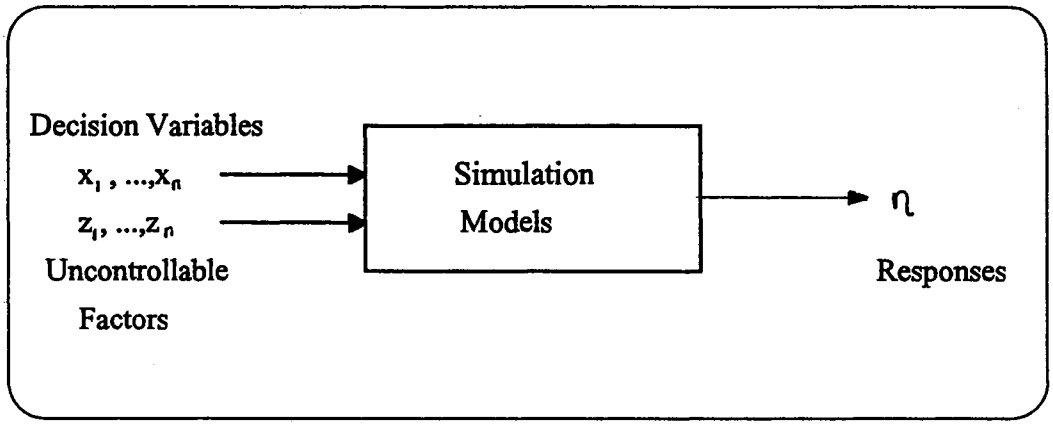


Figure 5. Input Output Relation

2. Observations of the system response at given points may contain random errors due to the effects of uncontrollable factors. The responses produced by the simulation runs will be

$$y_j = f_j(x_1,x_2,...,x_n) + \varepsilon_j \qquad j=1,....,m$$

where $\varepsilon_j = \eta_j - y_j$.

In recent years, the optimization of simulation models which combines multiple inputs and multiple responses has received increased attention. For example, consider an n-item inventory system and the simulation model for this system. Input variables are the reorder points and reorder quantities for each of these items and the responses are the average annual cost, average inventory level and average number of shortages. Determining the reorder points and reorder quantities which minimizes all three responses becomes a multi response simulation optimization problem. The interface between the optimization procedure and the simulation model is shown below (Figure 6- adapted from Biles and Swain(1983)).

The important characteristics which might affect the selection of multicriteria optimization techniques stated in Stuckman et al. (1991) are:

1. The number of decision variables;
2, The number of response surfaces;
3. The nature of the response variables;
4. Run time;
5. The ability of the decision maker to make preferences between various criteria.

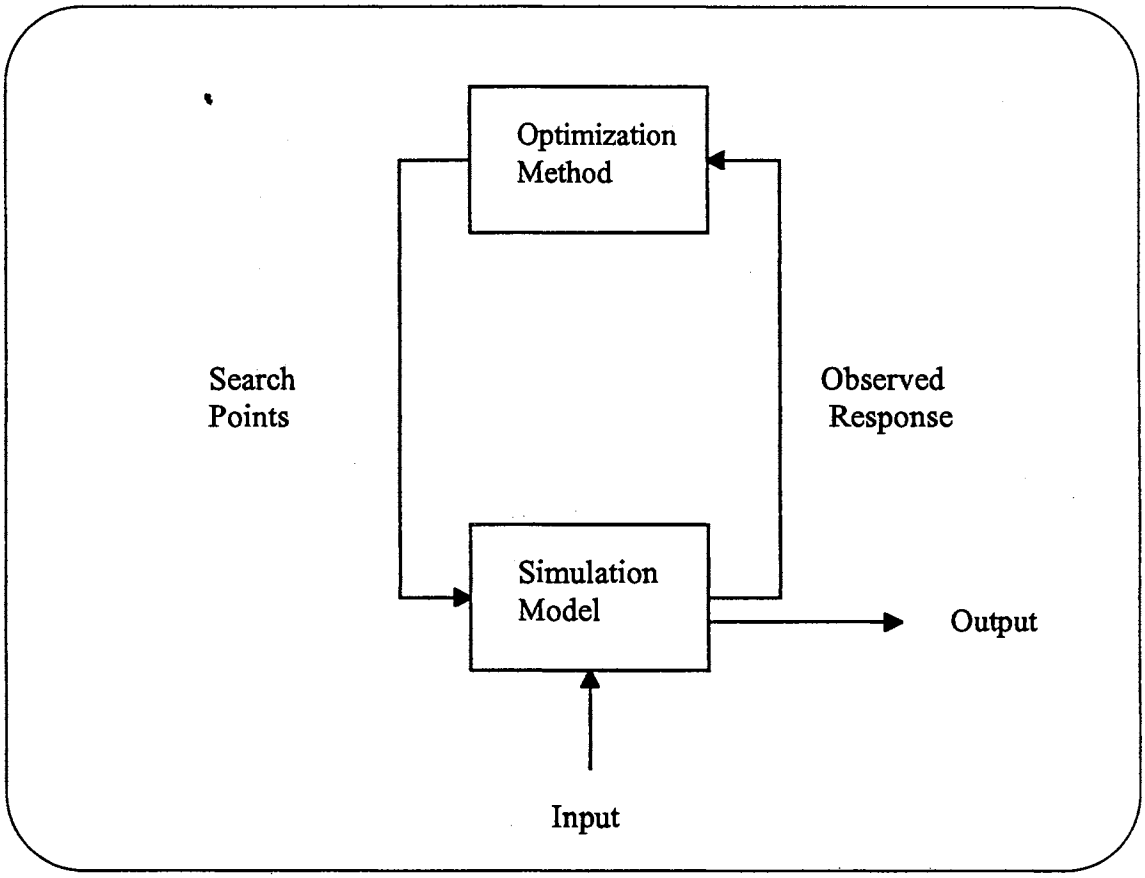


Figure 6. Simulation-Optimization Interface

Two approaches can be used to formulate a multivariable multiresponse simulation approach model.

1. Constrained Optimization : In this approach, one of the system responses is selected to be maximized or minimized while the other remaining responses serve as constraints within their prespecified bounds. Under this approach, the problem can be stated as

$$\text{Max(Min)} \qquad \eta_1 = f_1(x_1,\ldots,x_n)$$

s.t.

$$l_i < x_i < u_i \qquad i=1,\ldots,n$$

$$\eta_j = f_j(x_1,\ldots,x_n) \left.\begin{matrix}\leq \\ = \\ \geq\end{matrix}\right\} d_j \qquad j=2,\ldots,m$$

The bounds of the decision variables are usually known prior to the simulation runs. On the other hand, response functions are not known prior to the simulation and responses (f_j) must be estimated from the simulation runs. With this formulation, some of the responses that serve as constraints might be violated while the decision variables are kept within their boundaries. Another problem is that the random error might lead to wrong decisions relative to the constraints and relative to the objective function. For example, the analyst may believe that a given response is feasible when it is not. This problem can be prevented by employing appropriate variance reduction techniques.

2. Multiple Objective Optimization : In this approach, either responses are weighted to form a single objective function or treated like goals and used in goal programming.

One approach is to form a single objective function by assigning normalized weights w_i to each response before beginning the simulation. In general, the objective function combining n responses would be as follows:

$$\text{Max (min)} \quad W = \sum_{j=1}^{m} w_j f_j(x_1,...,x_n)$$

$$\sum_{j=1}^{m} w_j = 1$$

If s of the responses are to be maximized and (m-s) responses are to be minimized, the signs of the (m-s) responses should be reversed in the objective function. This approach is valid only when the responses are commensurable. If they are incommensurable, combining different types of goals into one is not possible and this approach cannot be used.

The second approach is based on the Geoffrion-Dyer interactive vector maximal algorithm and requires the user to adjust the sign and magnitude of the perturbations which specify the search direction. Then user is asked to choose one of the outputs which are presented in pairs. This approach provides the user with an entire view of the response surface. Details of this algorithm and its applications can be found in Montgomery and Bettencourt (1977).

A third approach is the goal programming approach. The goals are represented in terms of responses. For m system responses, the goals will be

$$G_j = f_j(x_1,...,x_n) \qquad j=1,...,m$$

Each goal must have an associated righthand side value which describes the minimum or maximum attainment level. Also each goal must be assigned a priority level P_j. P_1 usually represents the highest priority while P_m represents the lowest priority.

These weights are also called preemptive weights. Goals of higher priority levels are satisfied first, then the lower priority levels can be considered and lower priority goals cannot alter the goal attainment of the higher priority goals.

Goal programming does not attempt to optimize the given set of goals. It tries to achieve the most satisfactory level of goal attainment for all goals. Therefore the solution will not be an optimal solution, it will be a satisficing solution.

There has been many techniques applied in combining simulation and optimization. Although most of them are for single response problems, they are modified for multiple response problems. The techniques and methodologies used for multiresponse optimization can be classified according to the timing of the preferences, type of the preference information required, type of the decision variables (continuous, integer, mixed) and type of the objective function (linear, nonlinear, etc.). In all of the methods, the decision maker has to state his/her preferences by assigning weights, assigning priorities etc.

Graphical methods may be used to analyze the response surfaces. The surfaces are generated by simulation runs and are then plotted. The user then selects a point which he/she believes is the best choice. Montgomery and Bettencourt(1977) list the early works that used this approach. This approach may seem easy and attractive at the beginning, but as the number of variables and surfaces increase, it loses its appeal.

Another class of methods employs the techniques of constrained optimization. Lagrange multipliers are widely used for optimizing the primary response while the other responses are treated like constraints. Carrol (1961) developed a procedure which incorporates the constraint response into the primary response by means of a penalty

function. Heller and Staats (1973) used Zoutendijk's method of feasible directions to optimize the system. These techniques previously described use single criterion approaches. One disadvantage of these techniques is that the analyst must choose one response as the primary one. It is also difficult to perform a sensitivity analysis for the secondary constraints especially when there are small variations in the constraints .

Direct search methods which do not require the use of derivatives can also be applied. Typical methods are random search, Hooke and Jeeves pattern search, simplex search and Box's complex search.

Clayton, Weber, and Taylor (1982) used pattern search and gradient search for the optimization of multiresponse simulation models within the framework of goal programming with preemptive weights. This approach ignores the stochastic nature of the simulation, and is therefore good for simulation problems which are not stochastic in nature.

Biles (1977) used Ignizio's goal programming technique and described how first and second order designs make these techniques applicable to multiresponse simulation.

Box's complex search generates a set of starting points which satisfies the boundaries of the decision variables. A simulation run is performed at each of those points. Infeasible points which violate the response constraints are replaced with new ones and the worst response is chosen. The centroid of the remaining responses is found. A new point is found in the direction of worst point-centroid. Another simulation run is performed to check that this point satisfies all of the constraints. The procedure is repeated by adjusting the step size until the "best" solution is obtained or a predetermined number of simulation runs is reached.

Another approach is to use Response Surface Methodology (RSM). The methodology consists of taking starting observations according to the chosen experimental design. First-order designs (e.g. 2^k factorial or fractional designs) are used for first-order polynomials. A first-order model is fitted by least squares. Then the direction of improvement (steepest ascent or descent) is found and that path is followed until no further improvement is achieved. Another experimental design is applied and a new path is found. This process continues until the model shows lack of fit in the response. Additional points are then added to the search region to fit a second-order model. The selection of points is done with the help of second order designs. After the second-order surface is fit, a mathematical analysis is performed to determine the characteristics of the points (minimum, maximum or saddle points). According to the results, the analyst decides whether to continue or not.

RSM has been used in simulation optimization problems involving one response. Their use can be extended to multiresponse models. This could be achieved by applying goal programming concepts to the current procedure and each response can be associated with a priority. The response surface methodology would then be used to satisfy a single most important goal. After the highest priority goal is satisfied, an attempt is made to satisfy the second ranked goal without violating the high priority goals. The "optimal" solution is the satisficing solution which meets the goals in prioritized sequence. Rees, Clayton and Taylor (1985) used this approach to find the satisficing solutions to multiple response simulation models within a lexicographic goal programming model.

CHAPTER III

RESEARCH OBJECTIVES AND RESEARCH PLAN

Research Objectives

The main purpose of this research will be the development of an automated discrete-event simulation optimization system which not only would attempt to find an "optimum" solution, but also would attempt to determine the optimum "optimizing" technique(s) to be used in any situation. Specific research objectives are defined as follows:

Objective 1:

Build an optimization library module which will hold the different search algorithms. The module will be a collection of different search algorithms which may be suitable for different types of optimization problems. The module will provide automatic optimization of decisions with respect to an arbitrary user defined objective function that will be expressed in terms of the simulation output via a selected search procedure.

Objective 2:

Define the output analysis module which is necessary to analyze the simulation

results. This module will estimate the mean and variance of observations to provide statistical precision and accuracy to the experimenter and also control the simulation run length.

Objective 3:

Define the Executive Controller module. This module will provide the interactive communication between the user and the system and will deliver the necessary information between the simulation module and the other modules. It will also contain two structured logic- based submodules for the selection of the best optimization algorithm and detection of the degrading performance of the selected optimization algorithm.

Objective 4:

Develop the computer programs necessary to create an integrated environment for the proposed system. This will involve the coding of the search procedures, output analysis algorithms, rule based systems and other necessary interface requirements in FORTRAN.

Objective 5:

Demonstrate and validate the system developed. This will require the design of simulation experiments with analytical solutions and the investigation of the efficiency of the search procedures compared to the analytical solutions.

Research Plan

As mentioned previously, the purpose of this study is to develop an automated discrete event simulation optimization system which consists of an optimization library, an output analysis module, a simulation model, an optimizer, an executive controller and an output module. Figure 7 shows the relationship between the modules.

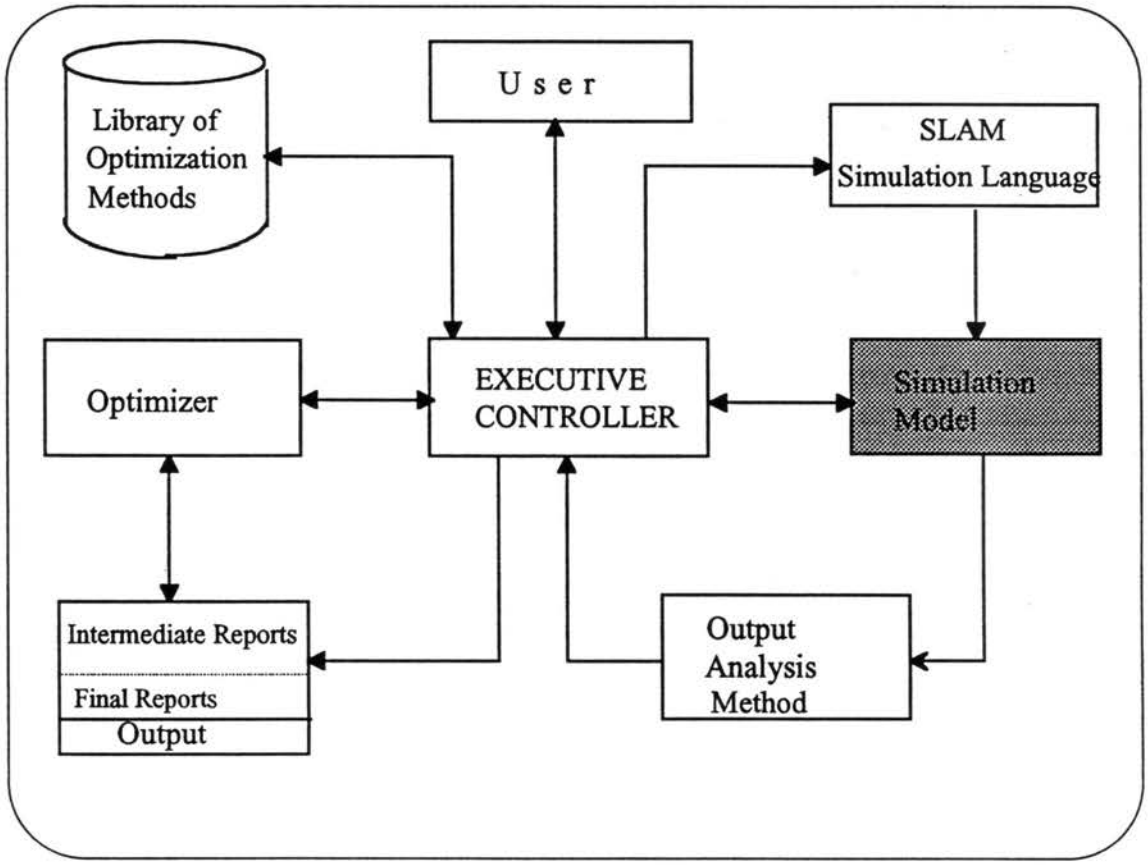


Figure 7. Automated Simulation Optimization System Modules

The only module provided by the user is the simulation module. The user has to have a complete simulation model and has to define the initial input values for the decision variables and the objective function in terms of performance criteria. SLAM was chosen as the simulation language for this study.

The research procedure will be discussed in terms of the phases required to achieve the research objectives and the specific tasks to be performed in each phase will be briefly explained .

Phase 1: Optimization library module.

Task 1 : Search the literature for currently available optimization procedures and evaluate the advantages and disadvantages of the proposed search methods.

Task 2 : Define the requirements for the selection of the methods that will be included in the optimization library.

Task 3 : Define the necessary interface requirements for the Executive Controller (EC).

One of the most important modules of the simulation optimization system is the library module which is a collection of different optimization methods. Each available optimization method must be evaluated according to a set of criteria to form the library. The selection process is based on effectiveness measures such as convergence, dependence on the starting conditions, initial variables reduction, capability of measuring nearness to optimum, execution time, and area of application. Also the search method should be constructed independently of any specific computer simulation and should be general enough to provide flexibility and wide applicability. Therefore the

methods developed for very specific problems should not be considered in this study. However, the way the executive controller is set up, it should allow the user to add his own program to the library module and invoke the program through the executive controller. In addition the user would not need to know the theory and mathematics underlying the techniques. Any technique which requires extensive knowledge of the theory and user interaction other than simple inputs would also be excluded. The main purpose of the library is to handle a wide variety of simulation models. The search procedures in the library can be used for single objective functions with real multivariables. The optimization module contains algorithms which allow both stochastic evaluation of the objective simulation models and direct evaluation of the independent variables from the simulation.

The search procedures which will be used in this study assume unimodality of the objective function. Since the algorithms are not designed to locate the global optimum of a non-unimodal function, the "optimum" was assumed to be the best of the local optima. The optimization problem was assumed to be unconstrained.

This module is controlled by the executive controller. EC invokes the optimization algorithm which is selected by the rule based system included in EC.

Phase 2: Output Analysis Module.

Task 1 : Search the literature for existing procedures and compare the efficiency and applicability of the procedures.

Task 2 : Explain the interface requirements with the simulation module and executive controller.

This module will analyze the output results and provide estimates of the system performance. The output analysis technique will be chosen from existing procedures to solve the initialization bias and the correlated time series problems. Since this study deals with nonterminating systems, the focus will be on the techniques for steady state simulations. In the analysis of steady state simulations, sequential methods are more preferable as mentioned in the literature survey. Among those, the batch means method is most preferred because of its simplicity and acceptance. Since this method requires the results to be free from initialization bias, a family of tests by Schruben for detecting initial bias will be used in the study. According to the test results , adjustments in the run length and warm-up period will be done either automatically or interactively through the executive controller.

Phase 3: Executive Controller

Task 1 : Define the user interface requirements.

Task 2 : Define the output analysis interface requirements.

Task 3 : Define the simulation optimization interface requirements.

Task 4 : Develop a structured logic for the selection of the optimization algorithm for a given system.

This module is the brain of the whole system. The Executive Controller (EC) contains rules for controlling all of the activity within the system. It provides communication among modules. It contains two rule based submodules for the selection of the optimization algorithm and detection of poor performance of the algorithm. It also provides interactive communication with the user. The user can define the objective function and can select the optimization method from the library or let the EC select the

appropriate method by using the structured logic developed for this purpose. The EC first asks the user to input the objective function, then asks questions about the variables and the system to determine which method is suitable for the system. After a method is chosen by the rule based system of the EC, the EC runs the simulation program. The results are analyzed by the output analysis module. If the precision is acceptable, the controller invokes the selected optimization algorithm (Optimizer) to update the parameter values. New values are passed to the simulation module by the controller. This process continues until the user's goal is satisfied or until the user-specified maximum number of runs is reached. The final results, along with standard SLAM output, are reported by the output module. The controller also has the capability of detecting the cases where the algorithm performs poorly. In this case the user is informed and given the option of choosing another algorithm or terminating the process. The user may get intermediate reports about the simulation, optimization or output analysis from the output module via the executive controller.

The response time of the system (finding the "optimum") is expected to be long considering the number of modules involved and the size of the system. It is unrealistic to expect the user to babysit the system all the time it is running. Therefore the EC module has to have a mechanism which lets the user interact at regular intervals and make decisions. The module will have time limits for user responses. If there is no response from the user within a given time limit, the controller automatically continues and uses default values if they are necessary. Figure 8 explains the logic flow of the proposed automated simulation optimization system.

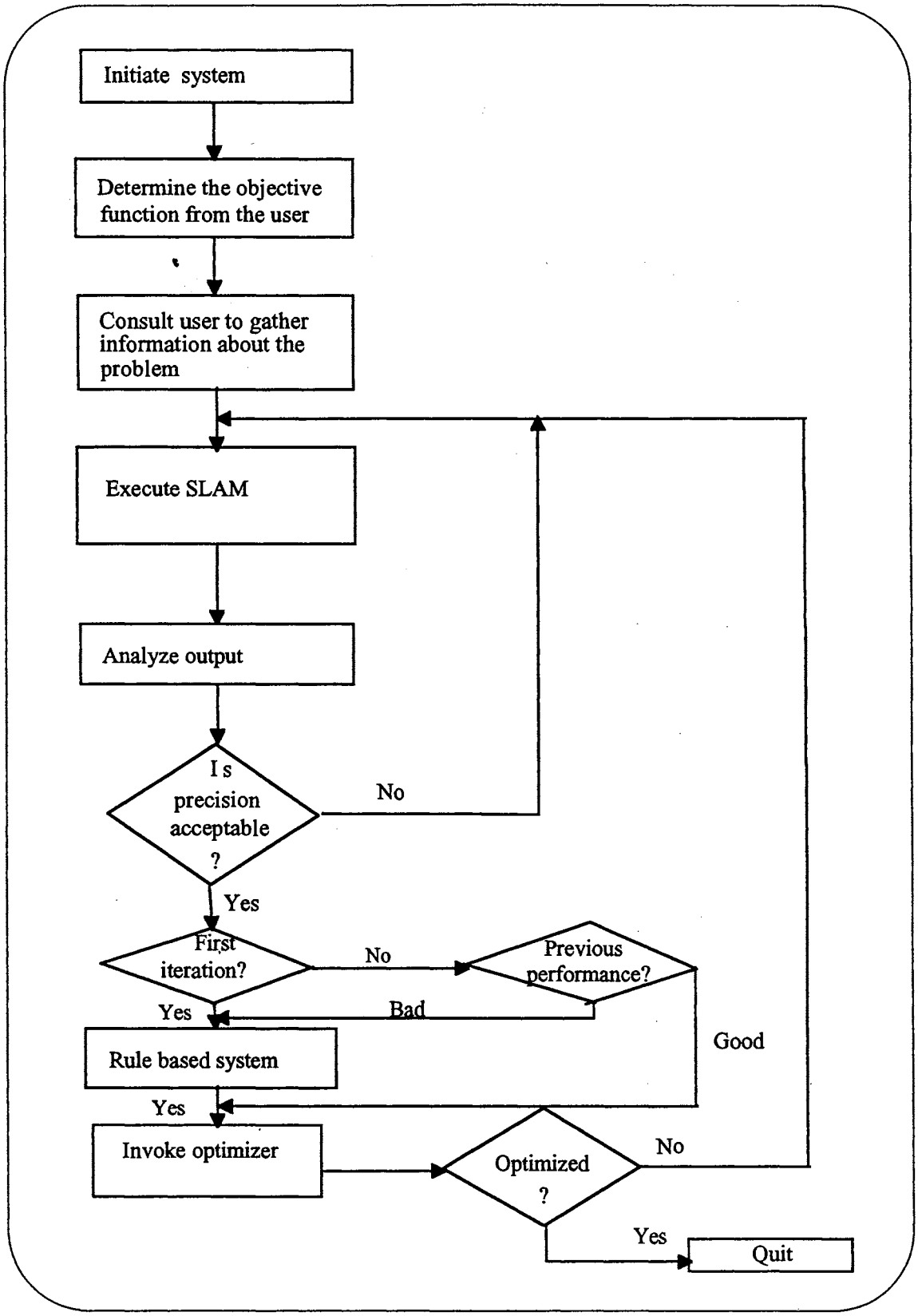


Figure 8. Flowchart of the Automated Simulation Optimization System

Phase 4: Software Implementation

Task 1 : Implement the optimization procedures of Phase 1 in FORTRAN.

Task 2: Implement the output analysis procedures of Phase 2 in FORTRAN.

Task 3: Implement the interface module in such a way that the system will be transparent to the user and be available on PCs.

Phase 5: Verification and Validation of the System

Task 1 : Define the measure of effectiveness for the search procedures

Task 2 : Design a factorial experiment which will explore different aspects of the model to test the proposed system under various conditions (e.g. system load, type of system). This will give a better opportunity to observe performance of the system rather than using one simple model which may or may not work for the developed system. The systems to be used in the experimental design may have an analytical solution or may be solved using some approximation techniques.

Limitations of the Research

As can be seen from the literature survey, there are many optimization algorithms which require different types of objective function, variables and constraints. Therefore there is a need to limit the type of the algorithms and problems to be used in this research.

The search procedures which will be used in this study require a unimodal objective function. If the characteristics of the objective function are unknown, there

will be no guarantee for the global optimum. The objective functions are also limited to unconstrained optimization. Single response, multivariable, real valued optimization problems will be covered in this study. Systems which may require the evaluation of integer variables are not considered in this study.

Since the simulation program is provided by the user, search methods which require alterations in the coding of the simulation program will not be included in the library module (e.g. Perturbation Analysis). The methods that demand extensive user knowledge of the underlying theory will also be excluded.

CHAPTER IV

RESEARCH METHODOLOGY

Introduction

The purpose of this study is to develop an automated discrete-event simulation optimization system. The system integrates simulation, optimization and output analysis techniques. The development of the system takes a user centered approach by addressing the needs of the user. This approach requires a design such that a user, a model and a simulation system can interact.

The automated simulation optimization system provides an environment within which the simulation model may be run, analyzed and optimized. The system must be designed to be as general as possible and must perform two major functions.

- Implement all application independent tools for the execution, analysis and optimization of the models (e.g. modules - output analysis, optimizer).
- Provide a specification for the interconnections and interactions between modules, model and the user (Executive Controller (EC)).

As can be seen from Figure 7, the system provides several modules for the control of execution , optimization and interaction which are invoked independently of the model provided by the user. The provision of libraries and modules for all execution

and analysis procedures reduces the effort of the user. The user does not need to consider any of the issues involved in the implementation of controls, statistical analysis or user interaction.

While designing the system, the following factors must be considered.

- Generality
- Completeness
- Suitability
- Flexibility

The generality of the system is defined in terms of the type of simulation system it is capable of analyzing. For example, it might be advantageous to allow models for both discrete and continuous systems or it may be practical to constrain the system to model just one class of systems. This research is limited to the latter case (discrete event, nonterminating simulations). Completeness and suitability are dependent on the objectives for the use of the system. The modules are designed with an open architecture which allows for the expansion and addition of more tools (e.g., new tools or tools for graphic visualization). Flexibility can be measured by the variety of models which can be analyzed and optimized successfully. The other desirable features of the system are ease of use, ease of communication, integrated environment, broad applications and reusable components.

These considerations led to the production of modular and reusable codes and created a framework within which models are executed, analyzed and optimized by catering to the user's conception of how he/she should work with such a system and

; the coding effort at all levels. Each module, its function and its submodules ıined in the following pages.

Optimization Library Module

Recently there has been an increasing trend towards the addition of analysis tools to the existing simulation languages. As the field of simulation has matured, the level of sophistication and performance expected in these languages has increased. Simulation users are demanding sophisticated graphical, mathematical and statistical tools to analyze and sometimes to optimize the complex systems.

This need suggests that a general library of simulation analysis tools which are independent of any simulation language should be created. This also coincides with one of the research objectives which is to build an optimization library for an automated simulation optimization system. The optimization library should contain various optimization algorithms to optimize many types of simulation problems. As was mentioned in the literature survey, there are many optimization methods available. Some of them are derived from nonlinear techniques and some of them are specifically designed for a certain type of problem. To include all the algorithms in the library module is infeasible and beyond the scope of this research. Therefore a selection process was developed to build the library module to optimize as many problems as possible.

The first criterion in the selection process should reflect the assumptions and limitations of the research. This study is limited to single response, unconstrained,

multivariable, real valued optimization problems. The main objective of the research is to create an environment which will help users at all levels to analyze the system and make decisions. To preserve the generality and flexibility of the system, algorithms that require alterations in the coding of the simulation program or extensive user knowledge of the underlying theory are excluded. This leaves methods that can be coded independently from the simulation model and do not require major user input other than initializing the variables, setting the parameters, etc. The mechanics of the algorithms will be transparent to the possible extent.

A general simulation optimization problem is given by

$$\text{Max(Min)} \quad f(x_1, ..., x_n)$$

If f(X) is a one dimensional vector, the problem reduces to the single optimization problem. If the elements of X are continuous variables, available stochastic search methods are more suitable. If they are discrete, integer programming techniques can be used.

This research is limited to search procedures with continuous variables and a single objective function. Methods for these problems can be classified as gradient based search methods, stochastic approximation methods, response surface methodology, heuristic methods and methods adopted from nonlinear programming techniques.

One may want to know which optimization methods can be applied to his/her model under what circumstances. Unfortunately, this question is difficult to answer in a satisfactory way and there is no right answer available directly . Each model has different properties and each problem poses different goals.

Unfortunately, there is no well established testing strategy for comparing optimization techniques for simulation optimization. Some tests are designed to validate a particular algorithm as an effective method. The most common practice is to have a set of comparative runs with a small number of alternative codes on a small set of test problems.

Some tests are performed on optimization codes for a very specific model form. Since the findings of such tests may have little value for other model classes, they should not be considered for evaluation of the techniques.

Ideally, one would like to have test functions which will represent the entire population of simulation model response functions. Unfortunately, there is no characterization of this population and a complete evaluation of existing optimization techniques. A partial solution to this difficulty is to search the literature for comparisons and look for comments made by authors on the performance of different algorithms.

Different algorithms are designed for solving different classes of optimization problems. Within each of these classes, different algorithms make specific assumptions about the problem structure. It is important to try to measure different algorithms using the same yardstick. Therefore comparison among algorithms should be based on a set of factors or effectiveness measures.

The proposed set of effectiveness measures contains the following factors (adopted from Bazaara and Shetty (1979) and Farrel et. al (1975)).

1. **Generality** - Generality of an algorithm refers to the variety of problems that the algorithm can handle. The restrictiveness of the assumptions required by the algorithm is also important. For example, a method developed specifically for the

optimization of the layout of an integrated circuit will violate the generality and will not be considered in this study.

2. **Availability** - Is the technique presently available? Is there a computer program or an algorithm which can be programmed?

3. **Reliability** - Reliability means that the procedure can solve most of the problems in the class for which it is designed with reasonable accuracy. Reliability is highly related to the number of variables. Therefore, one must consider the size of the problem when comparing algorithms based on reliability.

4. **Initial conditions** - Is the algorithm sensitive to the parameters and data? Is the technique dependent on initial conditions (starting vector, acceleration factor, step size, etc.). One would prefer that the algorithm not be very sensitive to the selected values of parameters.

5. **Initial variable reduction** - Does the technique eliminate unimportant variables or levels of input variables?

6. **Computational effort** - This measure is usually assessed by computer time, number of iterations or number of functional evaluations. One should keep in mind that computer time depends not only on the efficiency of the algorithm but also on the type of the computer used.

7. **Convergence** - The convergence of the algorithm is a highly desirable property. If convergence is certain for only some problems, what are they?

Numerous optimization techniques are mentioned in the literature survey and comments are made about their applications and performances. Representative

techniques for each class of optimization and the seven factors mentioned above are displayed in the matrix in Table IV with some brief comments.

Perturbation Analysis (PA) when applied properly, estimates all components of the gradients of the objective function from a single simulation experiment. Although this characteristic sounds very attractive, PA is severely limited by the class of queueing systems to which it can be validly applied. Although there have been recent efforts to eliminate these deficiencies, there is no clear cut characterization of other systems to which PA can be applied. PA assumes that perturbations are too small to change system performance. If the sequence of events that govern the system's behavior changes, the results obtained by PA may not be reliable. Considering the complex nature of most simulation models, this condition may not be satisfied most of the time. PA may also require considerable analytical work on the part of the algorithm developer. It may require some "customization" for each problem by expecting the modeler to have a thorough knowledge of the model and PA. In most cases, the modeler has to build the model from scratch and add additional code that is needed by PA. For these reasons and the sake of generality and flexibility, PA will not be included in the library of optimization methods.

Another unique class of optimization methods mentioned in the literature survey was Frequency Domain (FD) methods. FD methods depend heavily on Fourier transformation and require intensive knowledge of the underlying theory on the part of the developer. Although it is a promising research area, FD methods have some implementation difficulties. One practical difficulty in implementing FD for a large number of input factors is the assignment of driving frequencies to input factors so as to

METHODS	Convergence	Initial Conditions	Initial Variable Reduction	Compututational Effort	Availability	Comments
Heuristic Search	Unsure	No	No	Depends on simulation model & analyst	Yes	A function of analsyt experience; Requires no knowledge
Random Search	Unsure	No	No	Function of number of variables & simulation model	Yes- depends on existence of simulation	Very simple; Might be costly; Requires no knowledge
Hooke and Jeeves	No	Yes	No	Number of evaluations less than most methods	Yes	Algorithm may stop prematurely; Requires no knowledge
Nelder & Mead Simplex	No	Yes-No	No	Number of evaluations less than most methods	yes	Better than most derivative free algorithms

TABLE IV. Comparison of Optimization Methods

METHODS	Convergence	Initial Conditions	Initial Variable Reduction	Compututational Effort	Availability	Comments
RSM	Certain for unimodal surfaces	Important	In some cases	Highly variable; Depends on number of variables	Yes	Difficult for high order polynomials; Statistical analysis required
Simulated Annealing	Theoretically Yes	No	No	Strongly depends on number of variables & annealing schedule	Yes	Case dependent; Prior knowledge about system required
Perturbation Analysis	For some cases	No	No	Depends on number of variables and simulation model	Yes-limited for certain class of queues	Requires internal coding and knowledge
Frequency Domain (FD)	No	Yes	Yes	Depends on simulation model	Yes- used with other methods	Requires intensive knowledge about the underlying theory
Stochastic Quasi-Gradient (SQG)	Yes	Yes	No	Depends on number of variables and model	Yes	Converges very slowly

Table IV Continued. Comparison of Optimization Methods

avoid confounding effects of interest. Also within each run, FD methods require careful indexing of simulation generated observations together with sinusoidal variation of selected input variables. Such variation is usually difficult to arrange properly. Moreover, some of the basic theory supporting FD methods is incomplete. Considering the difficulties associated with the theoretical and practical aspects of FD methods, they will not included in this research.

Another method mentioned before was Simulated Annealing (SA) which is a style or strategy for solving combinatorial optimization problems. SA is a case dependent algorithm. The developer has to know the system very well and select the appropriate annealing schedule, initial temperature and temperature decay rates. These requirements prevent the SA algorithm from being reusable for different type of systems and makes it unsuitable for the optimization library. Also, as the number of variables increases, the efficiency of the algorithm is unknown.

Another group of methods mentioned in the literature survey was the stochastic approximation method. The stochastic quasi-gradient method based on Monte Carlo finite-differences tends to converge very slowly ($t^{-\frac{1}{4}}$). Two other stochastic approximation methods are basically of the same form: Robbins-Monro (R-M) and Kiefer-Wolfowitz (K-W), except that K-W uses finite-difference. Monte Carlo algorithms converge very slowly. The best possible convergence rate is of the order $t^{-\frac{1}{2}}$. This implies that one must multiply the run length by a factor of 100 to obtain an additional significant figure of accuracy. Glynn (1986) suggests that any algorithm which attempts to consistently estimate the gradient via Monte Carlo finite-differences will converge at a rate slower than $t^{-\frac{1}{2}}$. The R-M algorithm converges at a rate of $t^{-\frac{1}{2}}$

while K-W converges at a rate of $t^{-\frac{1}{3}}$. Two methods developed based on R-M are Common Random Numbers (CRN) and Likelihood-Ratio (LR) methods. It appears that the CRN gradient estimator developed in the discrete-event context is the Perturbation Analysis estimator. Therefore the discussion for PA is also valid for CRN. LR methods depend heavily on measure-theoretic probability. They have only been adapted to the regenerative method of simulation analysis and they are limited to the estimation of parameter sensitivities for Markov chains and Poisson arrival processes. It is unclear how this technique can be generally implemented in large scale simulation experiments.

L'Ecuyer (1991) explains the main techniques for estimating derivatives by simulation and surveys the most recent developments. He focuses on PA, Likelihood Ratios, finite differences and many of their variants. He also suggests that FD should be used when other methods would not apply or when they are judged too complicated to implement. He also mentions slow convergence of the FD method and the numerical problems associated with it.

In comparison to the new techniques for gradient estimation, the mathematical and statistical foundations of Response Surface Methodology (RSM) are not only more transparent, but also more developed. RSM can be applied to any discrete simulation, since it does not require manipulation or restructuring of the simulation code. Thus, from both a practical and theoretical standpoint, RSM possesses the advantages for gradient estimation and is a valuable addition to the optimization library.

The results of empirical investigations made by Smith (1973a) and Brooks (1959) indicate that a search algorithm based on RSM offers the greatest overall potential. As quoted from Safizadeh (1990) "Generally speaking, based on the existing

literature , response surface designs in conjunction with gradient based optimization techniques and search methods appear to best satisfy the optimization objective of the simulation".

Although the potential use of RSM has been discussed many times in the literature, the number of documented applications is limited. Smith (1973a) and Safizadeh (1990) comment that this situation may be due to the availability of simple, but brute force procedures and the unavailability of a cohesive, integrated presentation of RSM. A simulation practitioner would need to know about the mathematical and statistical bases of RSM techniques which could be found in statistical rather than simulation literature.

Therefore, an automated RSM program which requires a minimum amount of user input will be a valuable tool for the simulation optimization process. The concept and the steps of RSM will be explained in the following pages. The details of the method and the necessary calculations are presented in Appendix A.

Response Surface Methodology (RSM)

The goals of RSM are i) finding a suitable approximating function for the purpose of predicting future response and ii) determining what values of the independent variables are optimum.

The mathematical relationship between the response η and the input is the response function Φ and can be written as

$$\eta = \Phi (x_1, x_2, \ldots, x_n)$$

Since the exact form of Φ is unknown, a simulation model is being used to provide the response measurement from a specific combination of input variables. The response surface is approximated by a low-order polynomial (usually first or second order) called the graduating polynomial. The steps of the RSM procedure can be generally described as follows:

1. An appropriate experimental design is chosen to fit a first order polynomial which can be of the form

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

2. Simulation runs are made at the corresponding points.

3. Least squares regression is used to fit the points to the graduating polynomial.

4. The path of steepest ascent (descent) for the objective function is estimated.

5. A series of simulation runs are conducted along the path until there is no improvement in the observed region.

6. Steps 1,2 and 3 are repeated using the new region which is indicated by the steepest ascent(descent) path until the model stops providing a sufficiently good fit. The second order design phase is then entered.

7. The existing factorial design is augmented by additional points to form a second order design which permits estimation of quadratic effects, i.e., curvature of the response surface.

8. A second order polynomial is fit to the sub-region.

9. This polynomial is then analyzed by using calculus to determine the stationary point.

10. If needed, a canonical analysis and/or ridge analysis can be used to determine the nature of the stationary point. The flowchart in Figure 9 summarizes the RSM procedure.

The RSM program developed for this study will use fractional factorial for the first order design phase. A fractional factorial is a fraction of a complete factorial experiment. It can be particularly useful when the amount of experimentation required by the complete factorial is more than the experimenter can afford. Therefore fractional factorial fits well to our situation. The 2^{k-p} fractional factorial also has the added advantage of being able, by the addition of specific points, to evolve directly to a second-order design.

Considering the possibility of limited computer time and long simulation runs, RSM must use the smallest possible 2^{k-p} fractional factorial of resolution III. The resolution of a 2^{k-p} fractional factorial design is the length of the shortest word in the defining relation. The first-order designs to estimate coefficients in a first degree equation should therefore be of resolution of at least III (Box and Draper (1987)). This will ensure that no main effect is aliased with any other main effect. The only exception to this situation is when k=3. If the number of simulation runs or allocated computer time permits, full 2^3 factorial design will be used.

The central composite design will be used in the second-order design phase. Composite designs are first-order factorial designs augmented by additional points to allow the estimation of the coefficients of a second order surface. The central composite

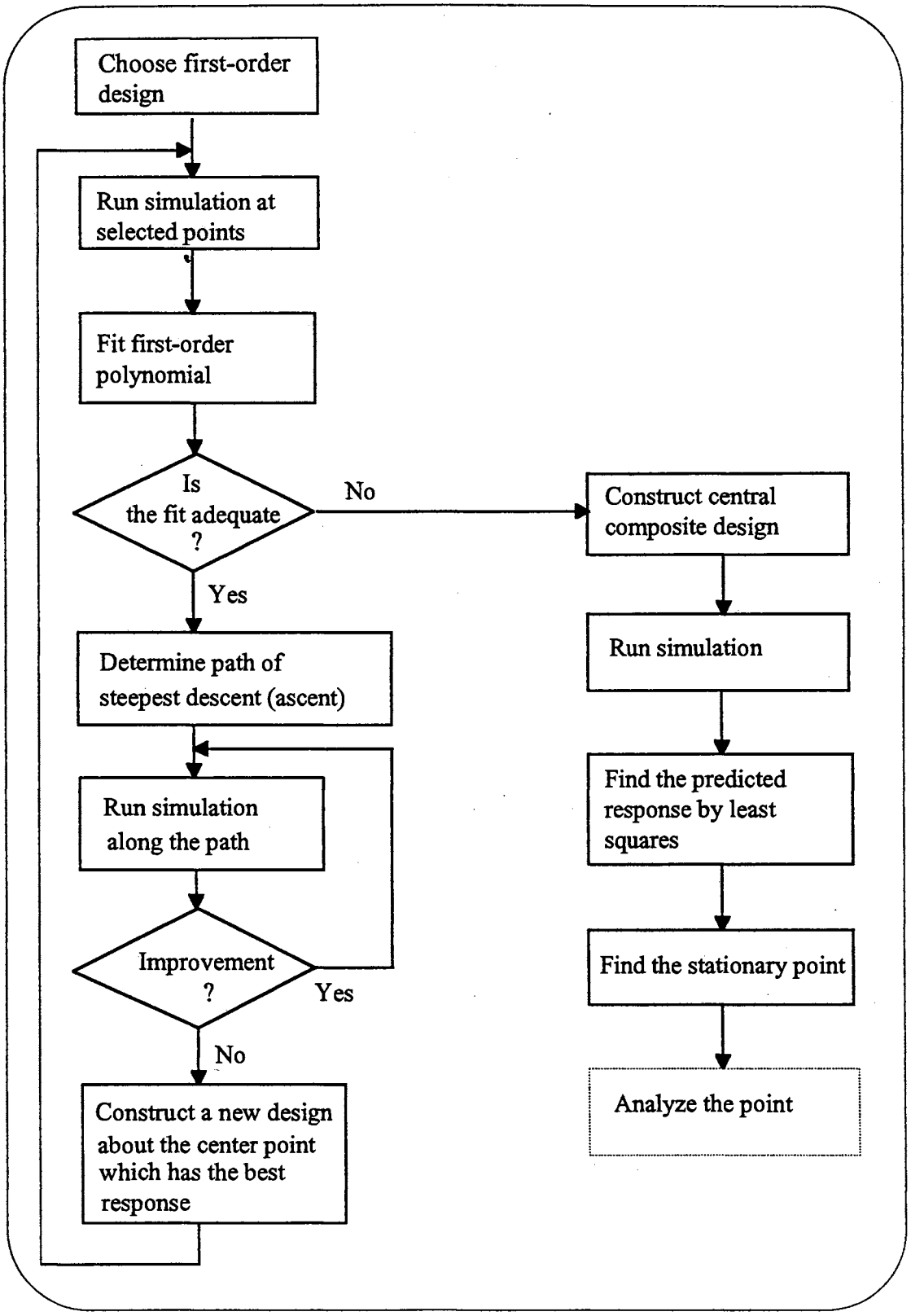


Figure 9. Logic Flow of RSM

design is formed by adding 2k axial points to the existing fractional factorial at its center point. This building block approach saves a number of usually costly simulation runs.

The points in the design are in the form of

$$
\begin{array}{l}
(0,\ 0\ ,\ldots,0)\\
(-\alpha,\ 0\ ,\ldots,0)\\
(\alpha,\ 0\ ,\ldots,0)\\
(0,-\alpha,\ldots,0)\\
(0,\alpha,0,\ldots,0)\\
\quad\vdots\\
(0,0,\ldots,-\alpha)\\
(0,0,\ldots,\ \alpha).
\end{array}
$$

Figure 10 shows a central composite design in three dimensions (k=3). The value of α can be chosen to make the regression coefficients orthogonal to one another, to minimize bias or to provide a rotatable design. A design is said to be rotatable when the variance of the estimated response is a function only of the distance from the center of the design and not of the direction. Thus the rotatable design is one for which the quality of an estimated response is identical for all points equidistant from the center point. This is useful when the experimenter does not know in advance where the center of the system will be or what will be the orientation of the system. The value $\alpha = (2)^{\frac{k}{4}}$ will be used in the RSM program to make the design rotatable. Further details for the least square method used for fitting designs and analysis of a fitted surface are given in Appendix A.

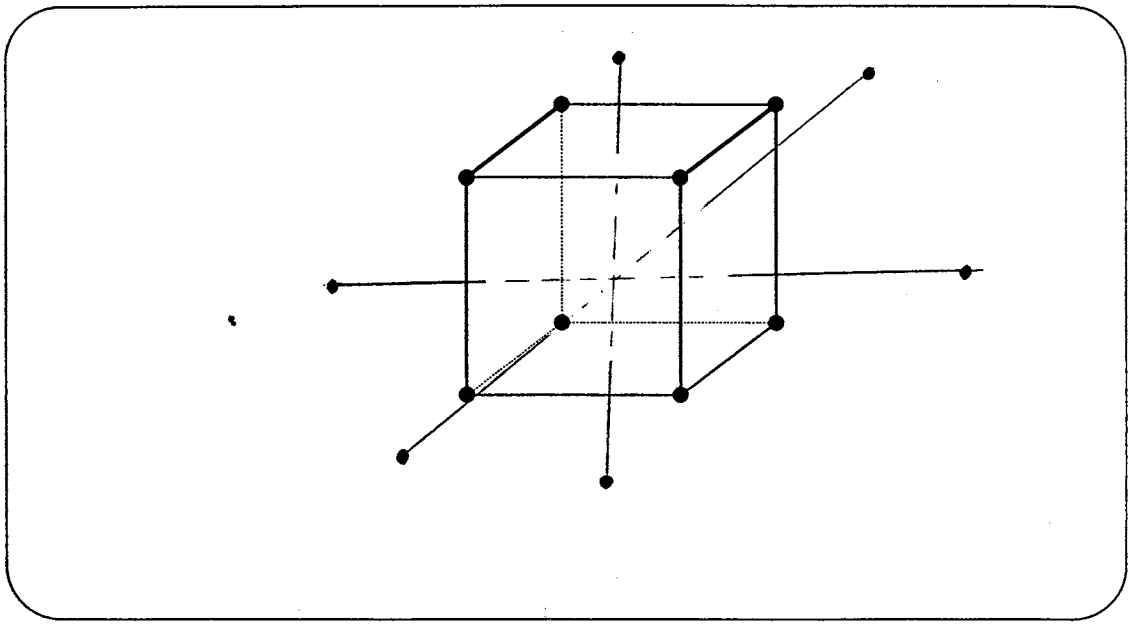


Figure 10. A Central Composite Design in Three Dimensions

Derivative-free Optimization Methods

The final group of methods to be considered for the optimization library is derivative-free unconstrained optimization methods. These methods are hill climbing methods, which determine the path toward an optimum by evaluating the objective function at several points rather than directly calculating derivatives. They are characterized by their simplicity, effectiveness and applicability to a wide variety of problems. One cannot single out one specific method to solve all the possible problems. "Unfortunately, there does not seem to be any one "best" method that can be

recommended, as a particular method might work very well on one problem and show up poorly on another" (Gottfried and Weisman (1973)).

From the comparison matrix , the Nelder & Mead Simplex method along with the Hooke & Jeeves methods show advantages. Although they assume deterministic function evaluations, both methods are robust and they need only few number of evaluations to determine the next setting of factors. They can be used in conjunction with other algorithms to produce better results. Thus, the optimization library will contain both pattern and path search methods.

Nelder & Mead Simplex Method:

The Nelder and Mead Simplex method for function optimization is a direct method that requires no derivatives of the function. Nelder and Mead's method is based on an earlier simplex sequential search strategy developed by Spendley, Hext and Himsworth (1962). In this method, an objective function in n variables is evaluated at the n+1 vertices of a general simplex. In two dimensions, the simplex would be a triangle, in three a tetrahedron. The simplex moves toward the optimum by moving away (reflecting) from the vertex with the worst value through the centroid of the remaining points. In two dimensions, this can be visualized as flipping over a triangle to move it down a hill. The method adapts itself to the local landscape using reflected, expanded and contracted points to locate the optimum.

Simplex reflections are expanded in the same direction if the reflected value is the best point; a poor value results in contraction. If the function value at the contracted point shows no improvement, the size of the simplex is reduced (shrinkage).

Figure 11 illustrates the steps of the Nelder & Mead Simplex for a function of two variables. PH, P2H, and PL denote the points where best, second best and worst function values occur, respectively.

Construction of the initial simplex and calculations for the locations of reflection, expansion and construction points are given in Appendix B. The details of the algorithm can be found in Nelder and Mead (1965).

The Nelder and Mead simplex method is widely used for simulation optimization, where the function it optimizes is subject to random noise (Barton (1992)). The algorithm is robust to small inaccuracies or stochastic perturbations in function values. This is because the method uses only the ranks of the function values to determine the next move not the function values themselves. Perturbations that do not change the rank of the values will have no effect on the algorithm's search trajectory.

In contrast to other optimization procedures, the simplex procedure approaches the optimum by moving away from the worst values of the objective function rather than by trying to move in a line toward the optimum.

The generality of the method has been illustrated by Olsson and Nelson (1975) in solving such problems as the direct maximization of the logarithms of a likelihood function, the solution of simultaneous equations, the maximization of a quadratic function which is subject to a quadratic constraint, the fitting of a line by minimizing the sum of squares of perpendicular distances from the points to the line, nonlinear least

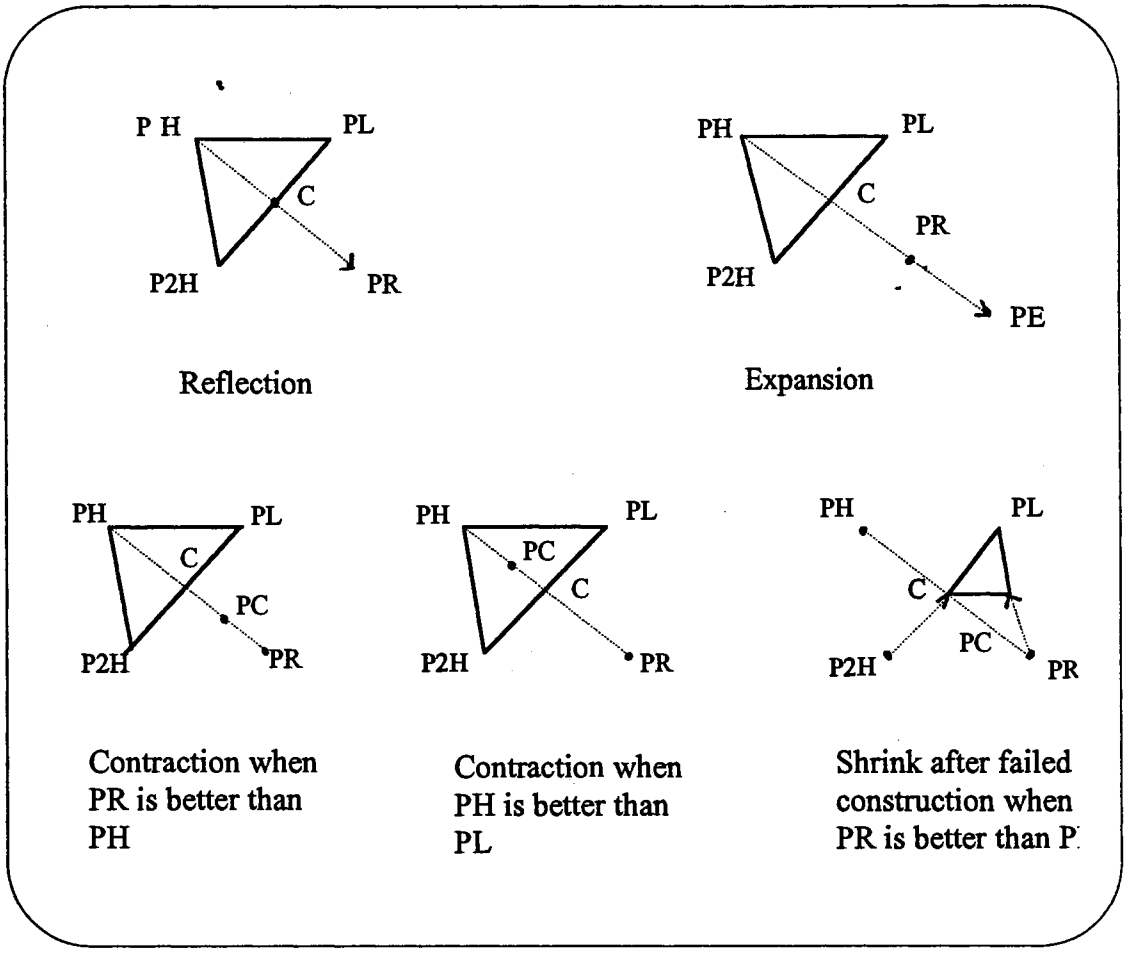


Figure 11. Nelder and Mead Operations

squares and the fitting of approximations to tabular data. Barton and Ives (1992) has also listed the various applications and comments on the large number of citations during the last 25 years. Birta(1977) has compared the simplex method with other optimization methods. He also showed the success of the method by applying different examples. According to the test results performed by Barton (1987), the Nelder and Mead method is the best overall performer among the nonlinear optimization methods tested.

Hooke & Jeeves Pattern Search:

The Hooke and Jeeves method is one of the most widely used optimization techniques. The method of Hooke and Jeeves (HJ) performs two types of search, exploratory search and pattern search. HJ is based on the idea that if a search strategy was successful in the past, then one should continue to move in that direction. The procedure alternates a series of exploratory and pattern moves. Figure 12 shows the types of search performed by the Hooke & Jeeves method.

The method starts by selecting an initial exploratory point and incremental values for each direction. It checks for an improved response at each incremented setting. The resulting improved setting becomes a new intermediate base point. The method then moves directly from the initial base point in the direction toward and through the new setting. This procedure continues until improvements changes cannot be made with the given incremental changes (step sizes). When this occurs, the step sizes are reduced and the procedure is repeated from the beginning. The presumed optimum is obtained when

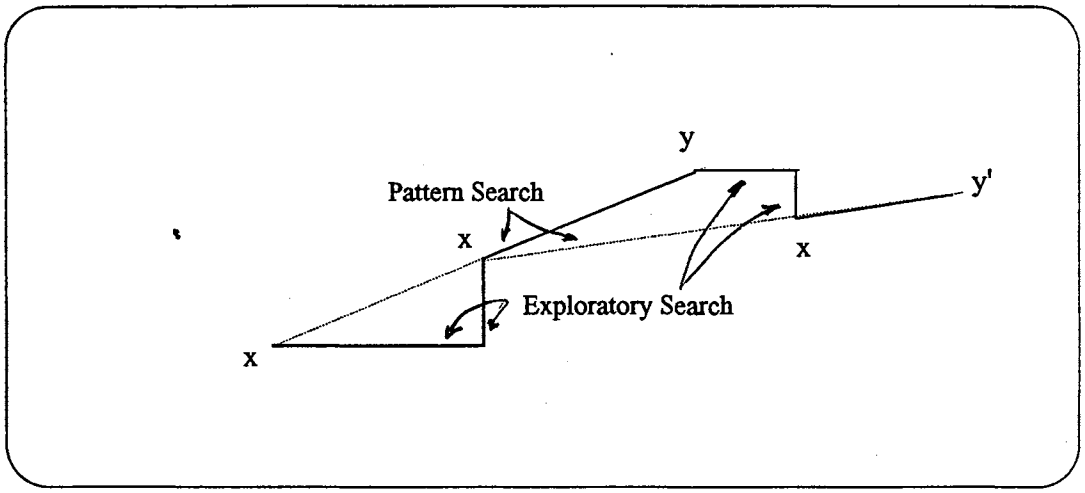


Figure 12. Exploratory and Pattern Search of Hooke and Jeeves

no search point yields an improved objective function value. The details of the algorithm can be found in Bazaara & Shetty (1977).

The Hooke and Jeeves method has been suggested for curve fitting and solving systems of equations. It is particularly well suited to functions exhibiting a straight, sharp ridge valley (Gottfried and Weismann (1973)). Although the procedure begins cautiously with short excursions from the base point, the step size grows with each success. Jacobson and Schruben (1989) have listed its applications. Pegden and Gately (1980) used the HJ method for automated optimization. Barton (1987) compared HJ with other techniques and showed its efficiency.

Simulation Output Analysis Module

While analyzing the output data, it is imperative to classify the simulation system into either a terminating or steady state type. The statistical tests for both types may differ and dictates the necessity for separate classification. Figure 13 shows the several submodules that are used to perform different types of analysis. Although Figure 13 contains submodules for both types, this research is limited to the steady state simulations. The submodules for terminating simulations are indicated as possible future additions to the module. Some of the submodules can be shared by both steady state and terminating systems (e.g., elimination of initial bias). The submodules may also be considered as a knowledge base combined with the executive controller. All of the submodules are expandable and they may contain more than one method. They are not dependent on each other and can be evoked independently. In case the user wants to remove initial bias from the system, he/she can be connected to the submodule BIAS or if he opts to use other methods, as long as he/she provides the necessary interfaces, he will be able to do so. The user also has the option of performing output analysis without optimizing the system. This structure enables even the most naive user to perform statistical analysis on the simulation system.

For a terminating simulation, the output analysis is fairly simple and straightforward, since the classical methods of statistical analysis can be directly applied. However this is not the case for a steady state simulation. It was seen that steady-state simulation output are more difficult to analyze, because the simulation practioner must address the problem of initial bias and the choice of the run length.

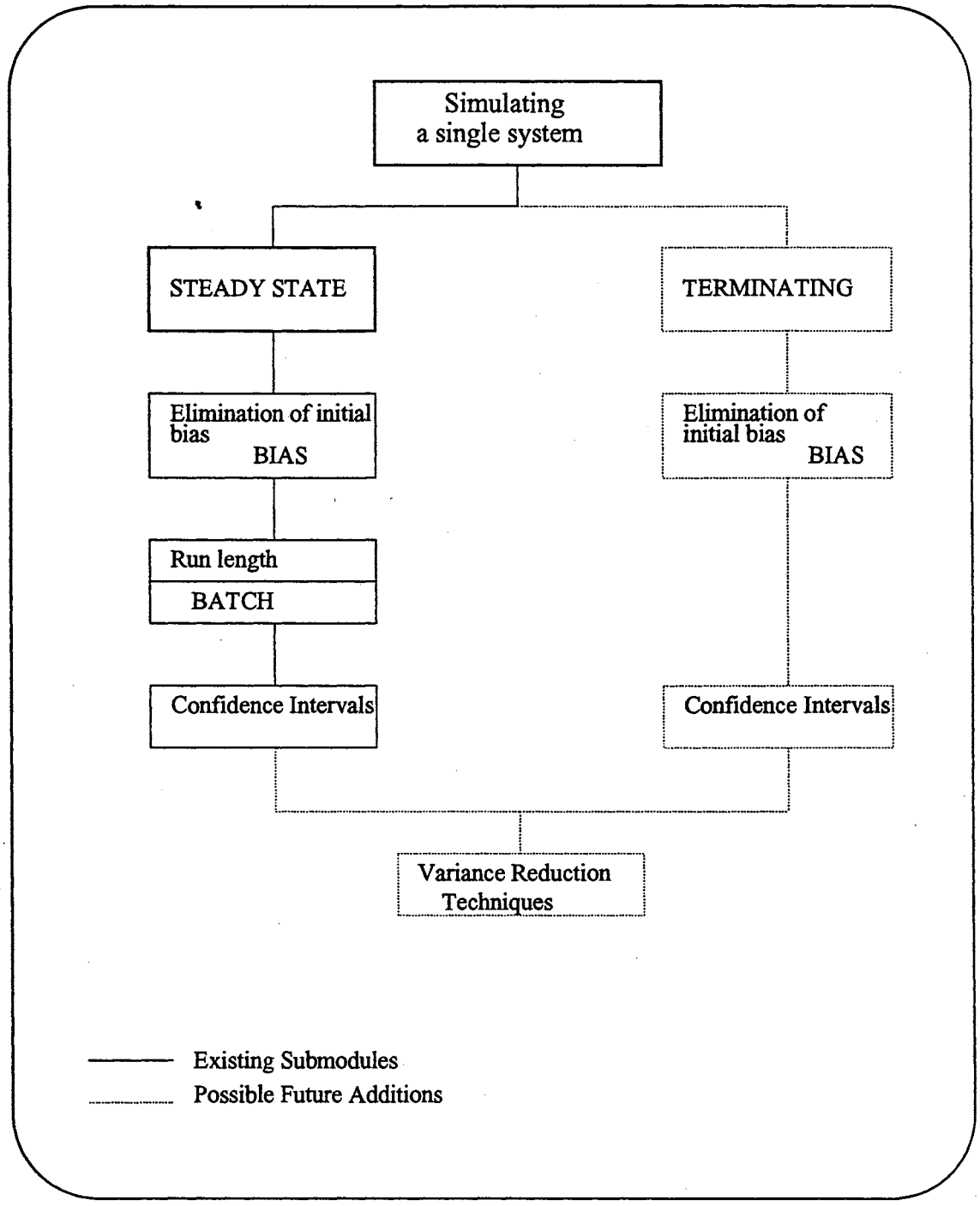


Figure 13. Structure of the Output Analysis Module

Elimination of Initial Bias

In order to evaluate a system's steady state characteristics, it is important to detect the existence of any initialization bias in the simulation output. The initialization bias problem and methods to eliminate initialization bias were discussed in Chapter II. Although some useful methods have been developed, "unfortunately there is no widely accepted, objective and proven technique to determine how many data to delete to reduce initialization bias to a negligible level" (Banks & Carson (1984)). Of these methods, the so-called optimal test developed by Schruben et al. (1983) appears to be the most powerful and robust in widely different situations (see Banks & Carson (1984), Banks, Goldsman and Carson (1990), Ma & Kochar (1993)). In a comparison study of tests for detecting initialization bias in simulation output (Ma & Kochar(1993)), the optimal test outperformed the other test when there was a significant bias. Based on these results, Schruben's optimal test was chosen to detect the initialization bias in this study.

The optimal test uses a hypothesis testing framework to detect the initialization bias. The null hypothesis is that the output mean does not change through the simulation run. The alternative hypothesis specifies a general transient mean function. In general, the null and alternative hypotheses take one of the following forms:

1. H_0 : No positive initialization bias exists in the observation sequence.

   H_1 : Positive initialization bias exists in the observation sequence.

2. H_0 : No negative initialization bias exists in the observation sequence.

   H_1 : Negative initialization bias exists in the observation sequence.

3.H_0 : No initialization bias exists in the observation sequence.

H_1 : Initialization bias exists in the observation sequence.

In the absence of any prior knowledge about the initialization bias, the third option should be used.

The procedure for the optimal test is as follows (see Schruben et al.(1983) for details):

1. Compute the sample variance (σ^2) and d (degrees of freedom) for the last half of the observation sequence, using the batch means method or autoregressive method.
2. Compute the following test statistic.

$$\hat{T} = \left(\frac{45}{n^3\sigma^2}\right)^{\frac{1}{2}} \sum_{k=1}^{n-1}\left(1-\frac{k}{n}\right)k(\bar{Y}_n - \bar{Y}_k)$$

where Y_k is the sample mean function defined as

$$\bar{Y}_k = \frac{1}{k}\sum_{i=1}^{n} Y_i \qquad k=1,2,....,n$$

3. Let α be the selected significance level of the test and $t_{(1-\alpha/2)}$ be the critical point from the student t distribution. The conclusion of a two-sided test for the initialization bias can be drawn as follows:

If $|T| > t(d, 1-\alpha/2)$, reject the null hypothesis of no bias.

Steady State Analysis

To analyze the steady-state behavior of the system, Law and Carson's sequential method based on the method of batch means is chosen. In sequential methods, the length of a simulation is increased until an acceptable confidence interval can be constructed. The advantage of the sequential methods is that if the technical assumptions of the method apply, the precision of the confidence interval is guaranteed. However, the length of the simulation is unpredictable. In spite of the difficulties associated with the sequential methods, they are a preferable approach to computing confidence intervals (Seila (1992)). Law and Kelton (1991) also recommend the use of sequential methods for the construction of the confidence intervals with a desired precision. Since, there has not been any approach which is proven to be more efficient as of this writing, Law and Carson's (1979) procedure was chosen for this research.

The procedure is based on the batch means method. The batch means method divides one long simulation run into a number of contiguous batches and then appeals to the central limit theorem to assume that the resulting sample means from each batch are approximately iid random variables. The method partitions the observations into a large number of small batches. Then it uses a "jackknifed estimate of the serial correlation" between successive batches and relative precision as a stopping rule. The jackknifed estimator was proven to be less biased for the estimation of lag i correlation (Miller (1974)) and is given by

$$\rho_j = 2\rho + (\rho_1+\rho_2)/2$$

where ρ_j is the jackknife estimator, ρ is the lag 1 estimator based on n batches, ρ_1 and ρ_2 are the usual lag 1 estimators based on the first n/2 and the last n/2 batches. Law and Carson's (1979) procedure divides m observations into 400 batches of size k. If the jackknife estimator between the resulting 400 batches is less than a threshold value, then the same m observations are divided into 40 batches of size 10k and the corresponding 40 batches are considered to be uncorrelated. Those batch means are used to construct a confidence interval if the ratio of the half length to the midpoint of the confidence interval is less than a specified relative width. If the estimated lag 1 autocorrelation is not less than the threshold value or if the actual relative width is not less than the specified one, additional observations are collected and above steps are repeated (see Appendix C and Law & Carson (1979) and Law & Kelton(1982a) for the details and steps of the algorithm).

Selection of Performance Measure : Since this study involved nonterminating simulation, it was necessary to decide a common measure of performance for the steady state checks so that we will know when to stop. Each model might involve a different measure which makes steady state analysis very difficult. For example, possible performance measures for a M/M/1 queue are the average waiting time in the system (W_s) or the average number in the system (L_s). On the other hand for a different system, to choose a performance measure may not be so straightforward (e.g. an (s,S) inventory system). However, every model will have an objective function and that function will depend on at least one stochastic measure. To preserve the generality of the system developed, to make steady state analysis simpler and to bring a practical

solution to the problem described above, the objective function will be used as the performance measure (e.g. C= $f(W_s, L_s)$). Since the main interest was to find the combination of decision variables which gave the best simulation response (objective), the individual behavior of the decision variables was not considered during the simulation run. One possible drawback of this approach is that the confidence intervals for some of the performance measures might be tighter than necessary. However this study is about the optimization of the simulation response and if the response is stabilized, so will be its components. One advantage of this approach is that since the user will provide the objective function it will be easier to understand and relate to the model.

Comparison of Two Systems : Law and Kelton's paired-t confidence interval method based on common random numbers was used to compare two systems (Law and Kelton (1991)). The same random numbers are used to simulate both systems. To ensure that synchronization occurs, different random number streams are dedicated to each activity and arrival. To perform the paired-t test at the end of each run, the performance measure was compared with the result of the previous run. If the systems are different, a counter for the redundant data search was kept and increased, but the search procedures continued independent of the outcome of the paired-t test.

Executive Controller

The last module of the system developed is the Executive Controller (EC). EC's task is to control all of the activity within the system. It provides a specification of the

interface between the simulation model and the system within which they run. It controls the model execution, optimization and interaction. This is the most crucial module of the whole system. Therefore its design is very important. To design this module, several questions must be considered. They are:

- What classes of system will be optimized?
- How can the model and data be efficiently connected to the system?
- How might the user interact with the model and the system?
- How can an optimization algorithm be chosen for a particular model?
- What should be the role of the simulation model and the language in this module?
- How can statistical analysis be connected to the model and the system?

As this short and incomplete list suggests, the design of the EC is not trivial. In designing the EC, one is forced to explore and define all relationships between the simulation model, users and system modules. As a result of these and limitations of the research, the EC may not implement all of these relationships in a way which is ideal for every application. But within the research scope, the EC was designed to perform as efficiently as possible.

The EC focuses on the interaction between all level of users, the simulation model and the optimization-analysis environment. Its design does not require the specification of a simulation language.

The EC supports the model and the user and forms a framework in which the simulation model can be executed, optimized and analyzed. It contains modular procedures designed to support analysis and optimization procedures. These procedures

include data specification and manipulation, input-output, intervention and interaction, model execution and optimization, pre- and post-execution activities, control of the analysis and a rule based system for the selection of an optimization algorithm.

The EC works with three major environments, model execution, output analysis and optimization. The user can work with the data associated with the model and the rest of the system through the menu system which is run by the EC. The menu system provides an organizational scheme, which can be understood by the user and the system, and a medium for effective communication between the user and the system and among the modules. The menu system removes parameter initialization from the model, provides a conscise and natural organization of the system understandable by the user and allows interaction with the system.

All the menus contain basically two parts: a default menu system and an additional menu system which is specific to the model and is created during the execution of the system. Using the menu system and giving options to the user provides a means of organizing, manipulating and storing data which is simple, consistent, efficient and comprehensible to the user. All the data can be input through the keyboard. Data sets may be created by various sources and stored on external storage devices. Data are transferred between modules through the files which were created during run-time.

Each module reads its data from an input file and writes the results to an output file which may become an input file for another module. Sample menus and screen printouts for the chosen cases are presented in Appendix D.

During system execution, the user can elect to interact with the running system at certain points. The user may want to see the intermediate results. At these points, the user has the option and control to terminate the system or change the system parameters if necessary. Current results can also be displayed on the screen or written to a file if the user chooses to do so. Due to the programming language limitations, the user does not have continuous access to the menu system and cannot interact with the system whenever he wants. This disadvantage can be compensated for by displaying the current status of the system, informing the user at regular intervals and giving the user a chance to intervene.

EC and Output Analysis Module (OAM):

Control of the execution of a simulation model is an essential part of the EC. The importance and necessity of rigorous and efficient methods in the analysis of simulation systems were mentioned in both Chapter II and in this chapter. As a result, an output analysis module was added to the system.

The simplest type of run control is manual, in which the user chooses a manual control option rather than an automatic control option. In that case, the user specifies criteria for the completion of the run. This criteria might be the number of simulation runs or the total allowed CPU time. In the case where the EC takes over, the user provided default values and was asked to specify the necessary parameters. After the verification of parameters, the simulation model is executed and the EC calls the OAM to

perform statistical analysis on the simulation output. First the BIAS submodule is invoked to check the existence of initial bias. If bias exists, the user is warned and advised to either increase the run duration, increase the warm-up period or both. If the user does not respond or does not know what to do, a very large number of observations are obtained and the warm-up period is increased batch by batch by, calling the BIAS submodule until the bias is negligible. If the output fails to pass the test, the program stops and displays a message which informs the user that under the given conditions the system will not reach steady state and the program must be terminated.

If the output passes the test, the OAM displays a message and calls the BATCH submodule. BATCH checks the observations to decide whether the desired precision of the confidence interval is achieved or not. BATCH increases the number of observations batch by batch until either desired precision of the confidence interval is reached or the simulation run length is reached. If the confidence interval is acceptable, the simulation response with the desired precision is written to the file for the next iteration of the optimizer. If the confidence interval is not acceptable, the user was warned by displaying messages. If the user does not increase simulation run length, the system terminates.

EC and Optimization Library Module

The EC also controls the selection of the appropriate optimization methods. A rule based system was developed to assist the user in selecting an optimization method.

After the collection of the methods for the optimization library, a structured logic consisting of a set of rules was used to decide which method should be used.

Various properties of simulation will have an impact upon the choice of the algorithm. Those properties are dimensionality, simulation run length and deterministicity. Since there is no or little information available about the form of the simulation response surface, dimensionality and simulation run length are the main concerns for this research.

The computer time required to optimize the system, (R), can be defined as

$$R = A(n) + nL$$

where A(n) is the computer time use by the optimization algorithm for n iterations and L is the execution time of a single simulation run. If L is small for a given system, R, can be minimized by minimizing A(n). This implies that it might be more effective to make more iterations than converge in fewer iterations. Alternately, if L is large, one would use substantial computational resources to minimize the number of iterations required by optimization procedures. These issues affect the selection of the specific method during run time. The dimensionality of the problems is limited to at most 10 variables in this study. The time required to perform one iteration of an optimization algorithm was assumed to be significantly less than the time required to run the simulation model. Since the number of variables directly affects the number of iterations, the decisions will be based on the number of variables (dimensions) and the simulation run length.

The rule-based system consists of a structured logic and if-then cases. If-then cases were based on the conclusions drawn from the literature, related examples and

especially, comments made by several authors. Figure 14 summarizes the selection process.

The selection process was performed in a series of questions and answers. The rule based system then evaluates the answers and selects the method. At the beginning, the user has two options;

1- user selects the method;

2- EC automatically selects the method.

If the second option is selected, the user is asked whether he/she has a priori information about the system behavior. If the user does not have any a priori information, the EC takes control from the user (at this point, the user had already entered the initial data) and executes the simulation model for a pilot run. Based on the simulation time of that pilot run, the EC implements the following logic:

- If there is only one decision variable, Hooke and Jeeve's pattern search is selected unless the user has added a single variable optimization algorithm (e.g. Fibonnacci Search) to the optimization library. Considering the contents of the optimization library, Hooke and Jeeve's pattern search can also be used for single variable problems. The following two methods are used for problems with two or more variables.

- Problems with less than 6 variables. Pattern search and the simplex method are known to perform poorly if the number of variables is large (6 or more). The simplex method is preferred to pattern search especially for simulation responses, since it requires only the rankings of the objective functions, not their values. If the computer resources (e.g. cpu time or number of runs allowed) are abundant or unrestricted, the

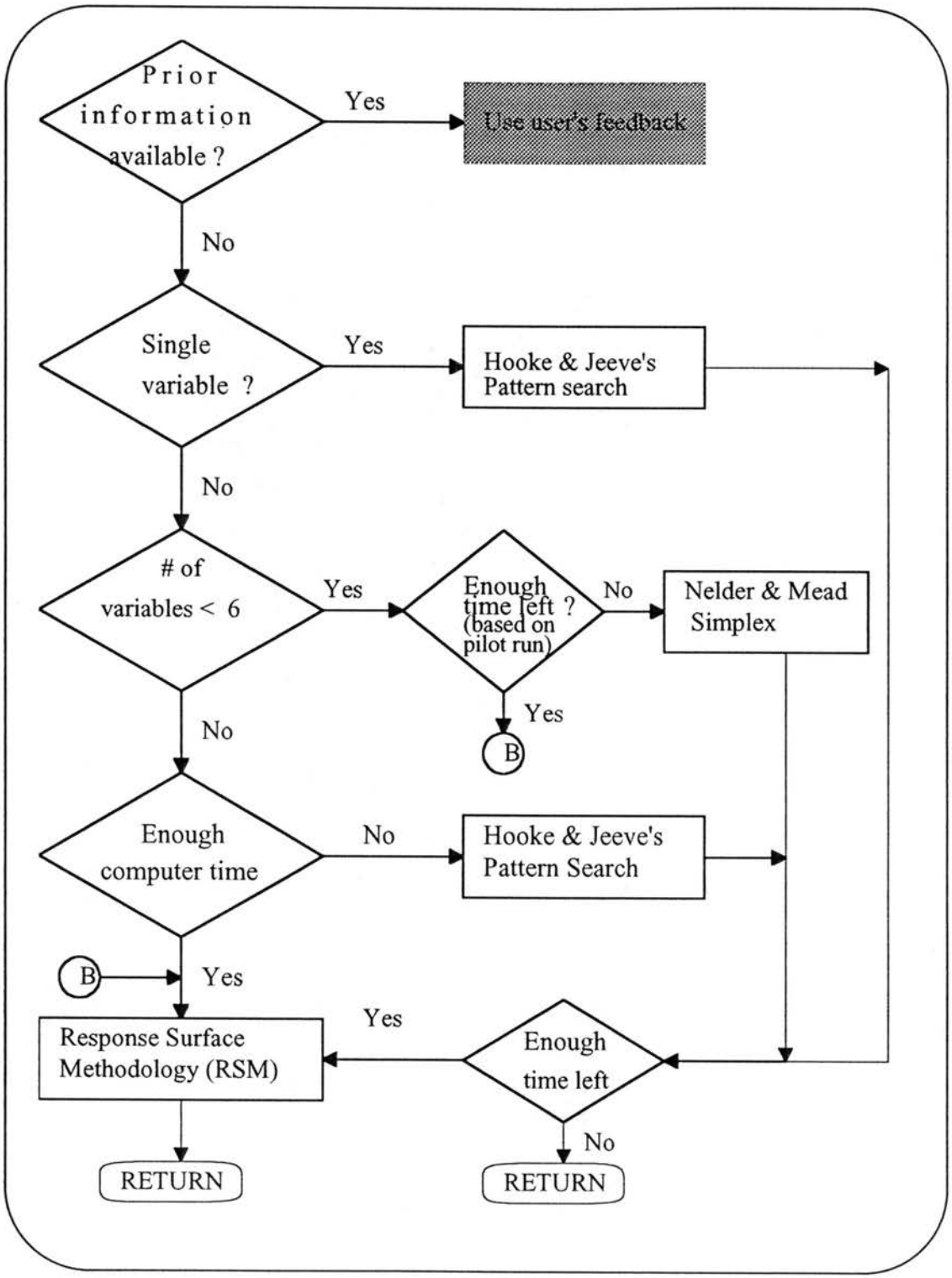


Figure 14. Logic Flow of the Rule-Based System

Response Surface Methodology (RSM) is selected. If there are only a few runs available (2*n) then pattern search is selected because of its robustness and its large step sizes at the beginning of the search. The purpose of this selection is to provide an insight about where the next starting point should be. In this case, although the user does not have an "optimum" solution, he/she would at least have a sense of direction concerning where it could be.

- Problems with 6 or more variables. RSM is selected as the optimization method. The user should be aware that a large number of runs will probably be necessary in order to construct the design points.

Once an algorithm is selected and successfully terminated, the EC checks the remaining computer time. If the remaining time is sufficient and the user wants the extra assurance and information, the EC calls the RSM from the point of the identified optimum. This practice reduces number of function evaluations and also gives the user an opportunity to explore the response surface.

The selection process described above assumes no prior information and feedback. If the user has previous experiences with the model and its behavior, the EC asks the following questions about the response surface.

- Case 1: possible ridges. In this case, pattern search is used due to its robustness and ability to climb ridges and perform well. The EC also has the ability to switch between methods. As an example , RSM turns on a switch to inform the EC of the existence of a ridge. When RSM fits the second order model and makes the simulation run corresponding to the stationary point found, if this point is out of the fractional factorial region, RSM raises a flag indicating the existence of a ridge and

returns control to the EC. In this case, there is need for a constrained algorithm. The EC might have switched from the RSM to the constrained version of pattern search, which might have used a deterministic objective function instead of using simulation. The coefficients of the objective function are determined by the second phase of the RSM and the initial starting point is the stationary point of the RSM. Due to the difficulty in finding a simulation model which would represent this feature and cause RSM to turn on the ridge switch and lack of constrained optimization methods, this feature could not be documented.

- Case 2: fairly flat surfaces. The literature points out that the RSM performs poorly in case of flat surfaces. This is due to trying to fit a quadratic equation to a flat surface. Therefore the EC advices the user not to use RSM.

- Case 3: irregular surfaces. The Nelder and Mead simplex performs reasonably well in case of irregular surfaces. Although this research assumes a unimodal response surface, not all unimodal surfaces have perfect "bowl" like surfaces. In reality, the simulation response surface may have an odd shaped valley and still be unimodal. Because of its ability to reflect, expand or contract, the simplex method tries to adjust itself according to the changing surface.

EC and Termination of Algorithms

The selected algorithm terminates under the following conditions:

1- Successful termination: The algorithm converged and found the minimum/maximum point within given limits.

2- Forced termination: The maximum number of iterations was exceeded and the algorithm was stopped prematurely.

3- Programmed termination: The algorithm progressed unacceptably slow, forcing the algorithm to stop early.

In case of successful termination, the EC displays the results. If there is no further request from the user, the system terminates. In case of forced termination, the EC displays the results and warns the user. The user either stops or restarts the algorithm from the best point found so far. In the last case, the progress made by the algorithm in the recent past is unacceptably slow, and the algorithm is stopped to be reinitialized or to be switched to another method.

The search algorithms, pattern and simplex search, each have a memory attribute. The action taken on the ith iteration depends on the information retained from previous iterations. Sometimes this retained information deteriorates in usefulness causing the algorithm to converge slowly or collapse. The recent past is taken to be the most recent 2n iterations (where n is the dimension). A programmed termination will take a place if

$$\frac{f(\bar{x}) - f(\hat{x})}{|f(\bar{x})|} < 0.5$$

where $\hat{x}$ is the current value of the decision variables, and $\bar{x}$ is the value of the decision variables at the beginning of the most recent 2n iterations in question (Birta (1977)). If the program detects slow convergence in the algorithm, it turns a switch on. The EC

warns the user and the user has the option of continuing, stopping the algorithm or switching to another algorithm.

Description of the System

Figure 7 showed the automated simulation optimization system modules and Figure 15 shows the logic flow of the system. The user inputs the data through the keyboard and the EC passes the decision variables to the simulation model, makes the first simulation run and selects the search algorithm. Simulation response is analyzed by the output analysis module, and passed to the optimization module to be used by the search algorithm. The search algorithm passes a new set of value of the decision variables to the simulation module. This process continues until one of the termination criteria is satisfied. All of the modules were implemented in standard ANSI FORTRAN except the simulation model which is written in SLAM. The programs were developed and executed on a 486-33 Mhz personnel computer. Although a mainframe would be more suitable for the program execution because of the lengthy and time consuming output analysis and optimization procedures, a PC was chosen for the increased portability and accessibility. Appendix H presents a user's guide of the system and Appendix I presents the computer program listings.

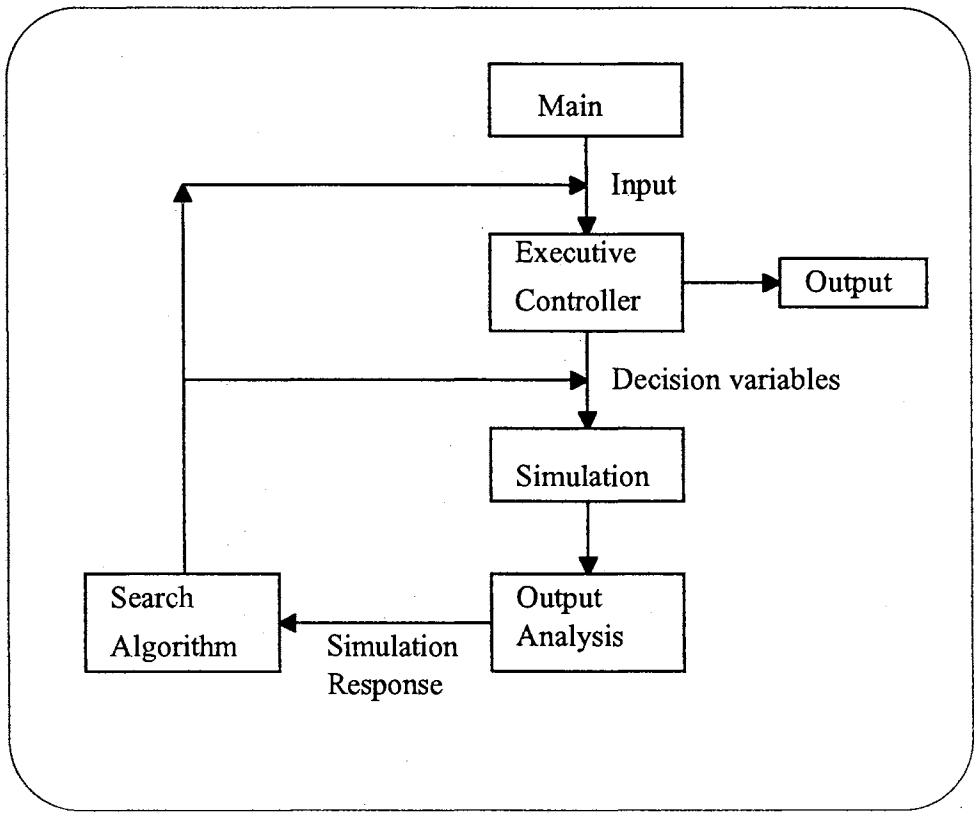


Figure 15. Logic Flow of the Developed System

Modifications to the Search Algorithms

Both pattern search and the Nelder and Mead simplex search revisit the previously evaluated points. Since simulation program uses the same random number seed for every run, the objective function value will be same for the revisited points. This causes redundant function evaluations and increases the program response time. Therefore, both algorithms were modified to reduce the number of simulation evaluations. Previously evaluated points and their corresponding function values were stored in a file. Every time when the simulation was called, the list of those points was searched. In case of a match, the corresponding response was used instead of calling the simulation. If there was no match, the simulation model was called and the list was updated. The overhead associated with the search and computations was significantly less than the length of the simulation run. For each algorithm, a counter was kept for the number of function calls saved.

CHAPTER V

EVALUATION AND TESTING OF RESEARCH METHODOLOGY

Introduction

This chapter presents the evaluation of the proposed methodology described in Chapter IV. The evaluation was done by implementing the proposed system and testing the system with selected problems. To test the system developed, a set of illustrative examples were chosen. This selection process was designed in a such way that various aspects of the system and a variety of the problems which might be encountered in real life were covered. To achieve this goal, queueing and inventory systems were chosen.

Queueing Cost Models

The first class chosen was a simple queueing system which had one server and arrivals and departures are Poisson with rates λ and μ. Figure 16 shows a simple M/M/1 queueing system where N is the capacity of the system. A queueing cost model was used to test the system. The objective of such a model is to find the level of service rate which balances the conflicting costs of offering a prescribed level of service (c_1) and the

cost resulting from the delay in service (c_2). Figure 17 shows these two costs as a function of the service level.

The first case of this class was a typical M/M/1 queue with a fixed arrival rate λ, an infinite system capacity and a controllable service rate μ. The objective was to determine the optimum μ based on the cost model consisting of the two conflicting costs explained above. In this case, the cost function is given by

$$TC(\mu) = c_1\mu + c_2 L_s = c_1\mu + c_2\lambda W_s$$

where

$TC(\mu)$: expected cost of waiting and service per unit time given μ

c_1 : cost per unit increase in μ per unit time

c_2 : cost of waiting per unit waiting time per customer

L_s : expected number of customers in system

W_s : expected waiting time in system (in queue + in service)

Although this problem was solved by using the simulation system, it also has an analytical solution and the optimum μ is given by (Taha (1987))

$$\mu^* = \lambda + \sqrt{\frac{c_2}{c_1}\lambda}$$

The second problem to be considered of this type was determination of the maximum number of customers that would be allowed in the system. With this restriction, the optimization problem became two dimensional. The cost function is given by

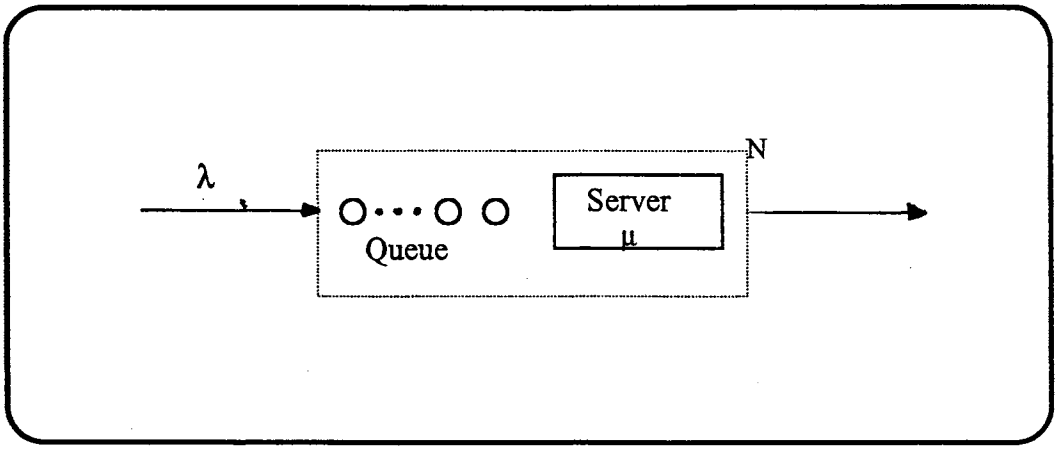


Figure 16 . A M/M/1 Queueing Model
(N is the system capacity)

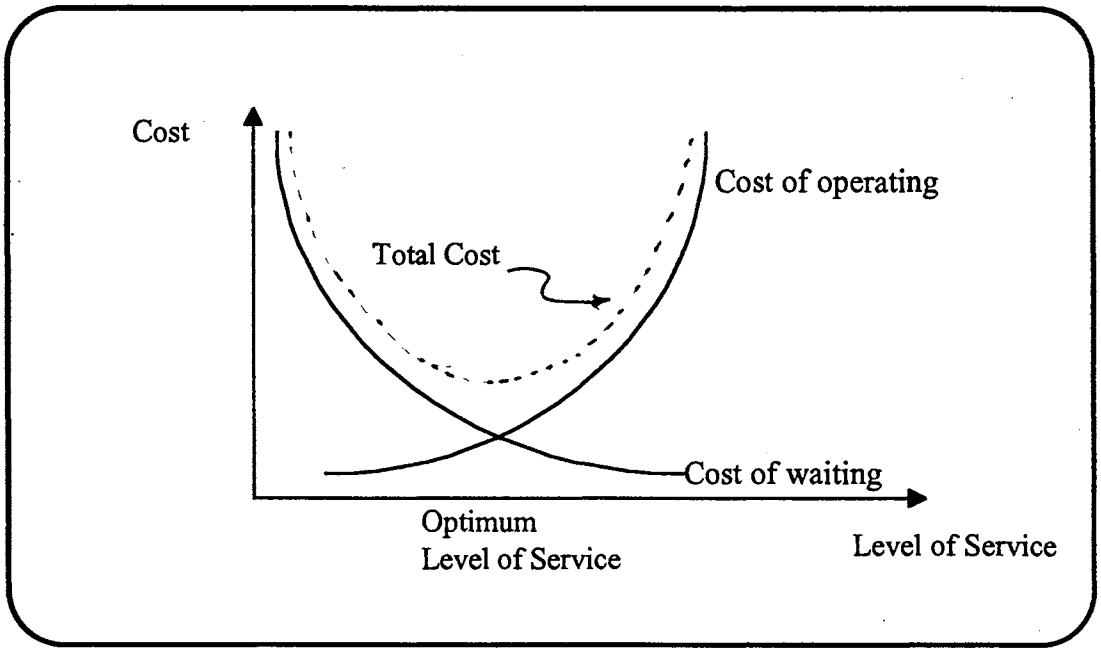


Figure 17. Total Cost as a Function of Service Level

$$TC(\mu, N) = c_1\mu + c_2L_s + c_3N + c_4\lambda p_N$$

where

c_3 : cost per unit time per additional accommodation unit

c_4 : cost per lost customer

λp_N : number of lost customers per unit time

Unfortunately, this problem does not have a closed form solution. Therefore, the optimum solution will be investigated by simulation and will be compared with the results obtained from a trial-and-error numerical solution. Figures 18 and 19 show the response plot of the M/M/1 queue with finite capacity. The graph of the cost function and Slam network statements are presented in Appendix E.

Continuous Review Inventory Models

The second set of examples included an inventory system which is frequently encountered in practice and simulation has often proven the only method of analysis. Inventory theory deals with the determination of the best inventory policy. Equations and models have been developed for setting parameters for specific situations. However, these equations are usually based on restrictive assumptions in order make analysis tractable. Also, a mathematical model of the inventory system may not include complex stochastic characteristics such as probabilistic demand quantity and/or lead times. However a simulation model can avoid such simplifying assumptions and include stochastic characteristics. In reality, demand is usually a random variable and so

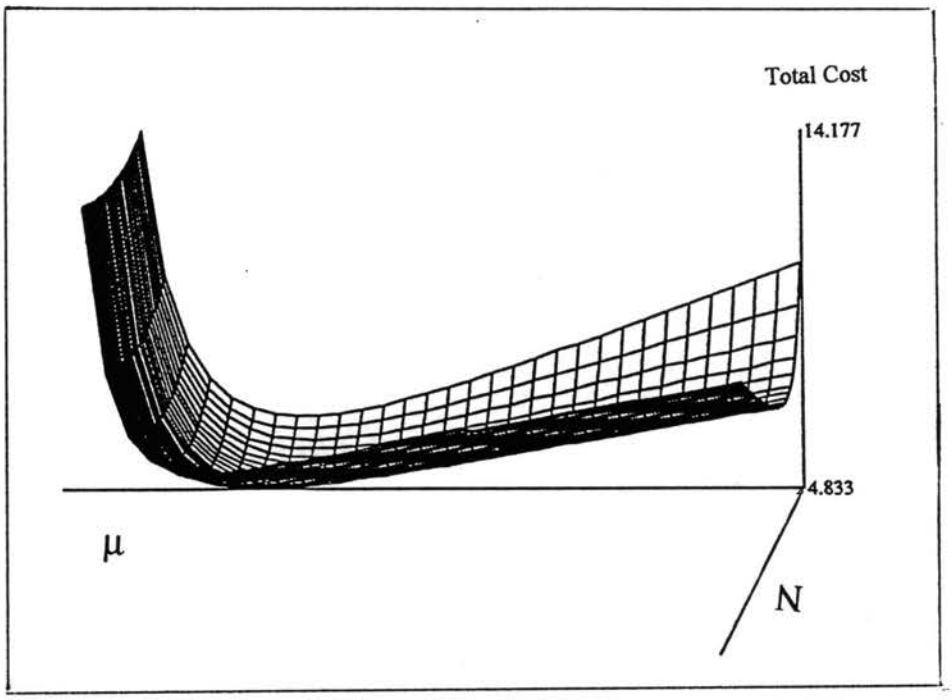


Figure 18. Response Surface for (M/M/1):(GD/N/∞) with $\rho < 1$

$1 \leq N \leq 30$

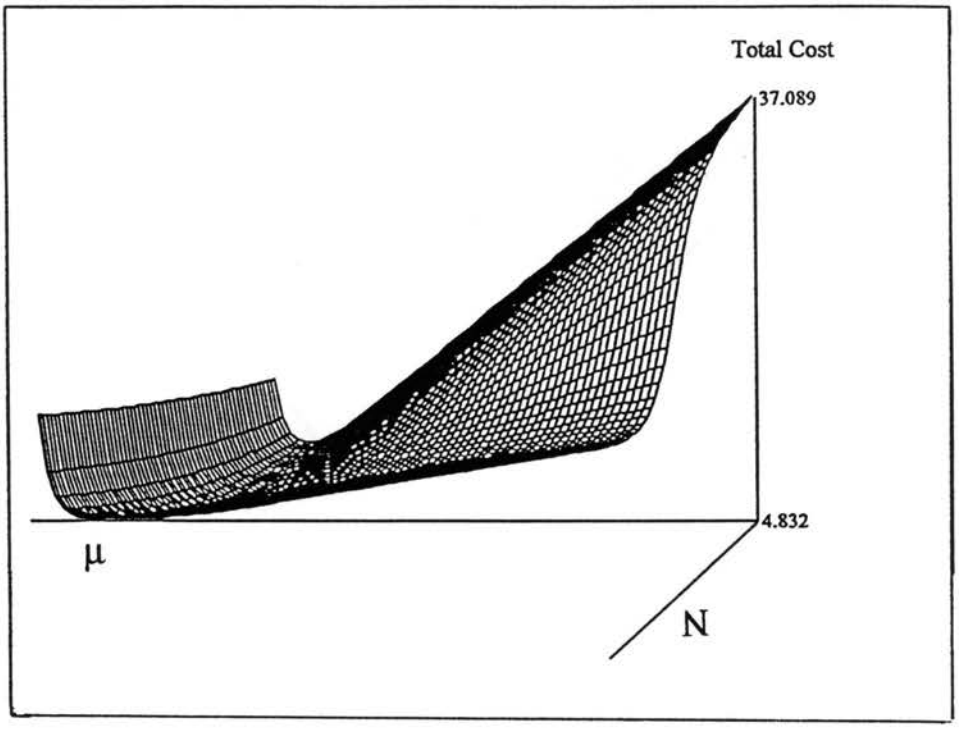


Figure 19. Response Surface for (M/M/1):(GD/N/∞) with $\rho > 1$

is the lead time. To include these dynamic realities, a continuous review inventory system in which the inventory was reviewed continuously and an order was replaced every time the inventory level reached a specified reorder point was chosen for this study.

The objective was to find the optimal reorder quantity and reorder level which maximized total profit by balancing the storage cost, reordering cost and shortage cost when demand quantity and lead times were random. The objective function can be expressed as

Total Profit = Total Revenue - Total Cost

Although it is not explicit in the objective function, the cost figures (e.g. shortage cost, holding cost, etc.) are functions of the reorder quantity and reorder point. Thus the decision variables for the inventory system are reorder point and the reorder quantity.

The complete definition and the graph of the cost function for the inventory system are given in Appendix F. Appendix G presents the Slam network and control statements. Since the objective was to find the long run average value for the measure of performance, the system was considered as a nonterminating system and it was analyzed and optimized with the techniques chosen for this study. Figure 20 shows the surface plot of the example problem presented in Appendix F.

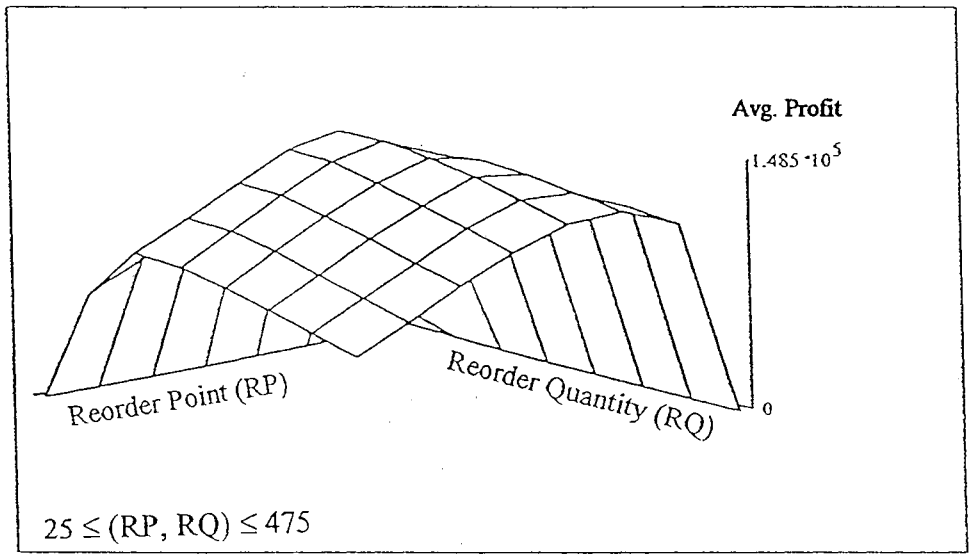


Figure 20. Surface Plot of the Continuous Review Inventory Model

Results

Table V presents the results of the pattern search for the first case. The M/M/1 queueing cost model is optimized by Hooke and Jeeve's pattern search. Since the shape of the function was assumed unknown, the algorithm was started at different points. Also the same starting point with different algorithm coefficients was used to observe the effect of algorithm coefficients. This cost model has an analytical solution with a minimum objective function value of 3.8003 with an optimum service rate of 43.37 (λ=30/hr). In every case, the algorithm converged successfully by finding the optimum service rate at 43. Algorithm coefficients, number of function calls, number of function calls saved due to the modifications in the search algorithms, number of redundant data searches (due to the output analysis routine) and total response time which is the time elapsed between the start of the algorithm and the termination of the algorithm are also displayed in the table.

Each simulation response was analyzed by the output analysis module for the initial bias and steady state checks. Since each iteration of the algorithm requires a different service rate, the truncation point was different for each run. But, in general, the truncation point was between 0 and 360 observations for the different points. To reduce the computational overhead and the response time, the model was initially warmed up before any analysis was performed. The truncation point for the warm up period was determined by plotting average batches and cumulative average batches

	STARTING POINTS				
	60	60	60	35	42
OBJECTIVE FUNCTION VALUE	3.782	3.798	3.7981	3.782	3.798
SERVICE RATE	43	43.3125	43.3125	43	43.3125
COEFFICIENTS S,A,B,T	1,1,0.5,0.001	1,1,0.5,1 E-05	2,2,0.5, 1 E-05	1,1,.5,0.001	1,1,0.5,0.001
NUMBER OF FUNCTION EVALUATIONS	33	47	45	23	30
NUMBER OF FUNCTION CALLS SAVED	5	11	18	8	12
NUMBER OF REDUNDANT DATA SEARCHES	4	6	4	4	5
TOTAL RESPONSE TIME (minutes)	117	154	152	179	87

Legend
S :initial step size
A :extension coefficient
B :reduction coefficient
T :termination criterion

Objective Function Coefficients:
$c_1 = 0.067$
$c_2 = 0.40$

TABLE V. Results of the Pattern Search for the M/M/1 Queueing Cost Example

(Bank & Carson (1984)) by deleting batches one at a time. This initial truncation did not affect the behavior of the algorithm. It only reduced the response time because the system reached the steady state sooner. For the given points and coefficients, it took two to three hours for the pattern search to complete the optimization process.

The second queueing example was optimized by the Nelder and Mead simplex method. The second case was an M/M/1 queue with finite capacity and the objective was to find optimum levels for the service rate and the system capacity to minimize the cost function. Since the system did not have a closed form solution, exhaustive search was done and the optimum values were found at μ=46.375 and N=8 with an objective function value of 4.83. The result of the Nelder and Mead simplex search for the second case is presented in Table VI. As can be seen from the table, the chosen method successfully converged and found the optimum values. The difficulty encountered with this case was that the time required for the system to reach the steady state was very long. Since the steady state performance measures were known (e.g. L_s , W_s), the results from the simulation model could be compared. Due to the nature of the system, a very long simulation run length was required to achieve steady state values. After experimenting with the system, algorithm coefficients for the BATCH submodule were fixed so that the system would run long enough and steady state values would be obtained. This precaution made the simulation time even longer. Also, since the BIAS and BATCH submodules both required continuous access to the files to read and write observations, the intensive I/O operations also increased the simulation run time significantly. A single simulation run took between 2-3 hours and the whole process, depending on the number of function evaluations, took between 3 and 4 days. Therefore, due to the

	STARTING POINTS			
	(44,6)	(50,16)	(35,12)	(60,10)
OBJECTIVE FUNCTION VALUE ($)	4.831	4.831	4.831	4.832
SERVICE RATE, SYSTEM CAPACITY	(46.27,7.8)	(46.3,7.8)	(46.47,7.8)	(46.6,7.8)
COEFFICIENTS S,A,B,C,T	2,1,0.5,2 0.001	2,1,0.5,2 0.0001	2,1,0.5, 2 0.0001	2,1,.5,2 0.0001
NUMBER OF FUNCTION EVALUATIONS	28	65	48	67
NUMBER OF FUNCTION CALLS SAVED	22	59	38	51
NUMBER OF REDUNDANT DATA SEARCHES	5	3	4	7

Legend
S :side of simplex
A :reflection coefficient
B :contradiction coefficient
C :expansion coefficient
T :termination criterion

Objective Function Coefficients:
$c_1 = 0.067$
$c_2 = 0.40$
$c_3 = 0.10$
$c_4 = 0.80$

TABLE VI. Results of the Simplex Method for the (M/M/1):(GD/N/∞) Queue

long simulation runs, a limited number of runs were done. Only 5 starting points were tested. For all the cases , the same algorithm coefficients were used.

Table VII shows the result of the RSM algorithm for this case. Although the initial intention was to demonstrate RSM for each of the different cases, due to the enormous simulation response time, RSM was used only once. For that run, the number of replications was set to one. There were two reasons for this. One obvious reason was the time considerations. The average response time for multiple replications would be at least 10 to 15 days. The other reason was that steady state measures were used in the method. Each simulation run was tested for steady state conditions. Even if there were more than one replication, their results would be very similar and the error term would be very small. Although two results might be statistically different, it was judged that the difference between two runs would be insignificant and acceptable from a practical point of view. Table VII summarizes the RSM algoritm which was started at (44,6).

The last case, a continuous review inventory model, was optimized by the Nelder and Mead simplex algorithm. The results are shown at Table VIII. Since this model did not have a analytical solution, an exhaustive search was made to establish the optimal solution. Function values for the response surface were obtained by executing the simulation for long period of time. Each alternative system was started with an empty inventory to introduce an initial bias to the system. Although Figure 20 shows a single maximum, the response surface is possibly multimodal. Due to the unknown characteristics of the function, different starting points were tried. Generally the average total profit value was increased due to the removal of initial bias. The program detected

Run	Service Rate	System Capacity	Response	Comment
1	44	6	5.020	
2	43	5	5.356	
3	45	5	5.240	
4	43	7	4.931	Fractional Factorial Completed
5	45	7	4.872	Explore the path
6	44.47768	8.2215	4.846 *	
7	44.734	9.4137	4.873	
8	44.99	10.606	4.941	Seek new path
9	43.477	7.721	4.877	
10	43,477	8.721	4.866	
11	45.477	7.721	4.838*	Fractional factorial completed
12	45.477	8.721	4.847	Explore the path
13	62.083	7.839	5.367	
14	78.534	7.481	6.312	Second order design phase
15	44.477	7.514	4.857	
16	44.477	8.928	4.858	
17	43.063	8.221	4.878	
18	45.892	8.221	4.835*	

* - optimum so far

Table VII. Results of the RSM

	STARTING POINTS				
	(150,150)	(100,100)	(100,100)	(200,250)	(250,400)
OBJECTIVE FUNCTION VALUE ($)	149,567.0	149,300.00	150,767.00	147,980.00	147,090.00
OPTIMUM REORDER PT. REORDER QUANTITY	(186,167)	(182,174)	(186,123)	(158,261)	(153,320)
COEFFICIENTS S,A,B,C,T	75,1,0.5,2 0.001	50,1,0.5,2 0.0001	75,1,0.5, 2 0.00001	50,1,.5,2 0.001	50,1,0.5,2 0.001
NUMBER OF FUNCTION EVALUATIONS	58	31	42	35	42
NUMBER OF FUNCTION CALLS SAVED	1	30	38	32	44
NUMBER OF REDUNDANT DATA SEARCHES	9	1	4	0	2
TOTAL RESPONSE TIME (minutes)	698	428	412	358	364

Legend
- S :side of simplex
- A :reflection coefficient
- B :contradiction coefficient
- C :expansion coefficient
- T :termination criterion

Objective Function Coefficients:
- PPU = 10000
- CPU = 5000
- CPO = 100
- HC = 100
- CLS = 200

TABLE VIII. Results of the Simplex Method for the Continuous Review Inventory Model

the "optimum" point at Reorder Point (RP) =186 and Reorder Quantity (RQ) =123 with an average profit of $150,067.

Thus, the last objective of the research was accomplished by demonstrating the developed system for different classes of problems.

CHAPTER VI

SUMMARY, CONTRIBUTIONS and RECOMMENDATIONS

Introduction

This chapter includes a summary of the research, contributions of the research and recommendations for future research in the area of simulation optimization systems.

Summary

The goal of this research was to develop an automated discrete event simulation optimization system. The research can be summarized in terms of the research objectives that were accomplished. The first research objective was to build an optimization library. For this purpose, an extensive literature search was made. A selection process based on the limitations of the research and a proposed set of performance measures was developed. The details of the selection process are presented in Chapter III. Since there was no single optimization method which performed well on all given functions, more than one optimization method were selected. The optimization library contains deivative free methods (Nelder and Mead simplex method and Hooke and Jeeve's pattern search method) and a gradient estimation based

method (Response Surface Methodogy (RSM)). The details of these methods were presented in Chapter V.

The second objective was to build an output analysis module. The output analysis module contains two methods. After searching the available literature (see Chapter III), Schruben's optimal test was chosen to detect the initialization bias in the simulation output. Law and Carson's sequential method based on batch means was used to analyze the steady state behavior of the system. The results were tested for initial bias and if any bias existed, observations were truncated, batch by batch, until there was no bias left or no observations left. If the removal of bias was successful, the remaining output was analyzed with the batch means method. When the desired precision of confidence interval was obtained, the performance measure was computed and sent to the optimization module as a next set of input data to the optimization method.

The third objective was to develop the Executive Controller (EC) module to control the activities of the system. The EC controlled model execution, output analysis and optimization. A rule based system was developed to determine which optimization routine was appropriate for a given problem. The EC also detected any changes in the system and the performance of the algorithms. The communication between the user and the system and modules was achieved with inputs through the keyboard and direct access files. All inputs and outputs were written to the files and read from the same files when it was necessary to do so.

The fourth objective was achieved by developing the necessary computer programs to create the system described in this study. For this objective, a modular computer program was developed to find the optimum solution to multivariable problems

simulated on a computer. All of the implementations were done in FORTRAN except the simulation program which was implemented in SLAM. Program listings are given in Appendix I. A user guide was also provided for the user and presented in the Appendix H. The user guide describes how to use the system, input-output requirements, limitations of the programs and necessary user modifications in case of any change in the current structure of the program.

The developed system was demonstrated and conclusions were drawn about the system efficiency. The testing and evaluation of the system was accomplished by selecting two different queueing systems and a continuous inventory problem. Each problem was optimized by the method which was selected by the rule based system of the EC.

Contributions

By reviewing the accomplishments of this research, it can be concluded that the objectives of this study have been achieved. The achievement of the research objectives provides the following contributions:

- Development of an integrated system of optimization techniques with simulation and statistics.

- Development of an environment within which simulation model could be run, analyzed, and optimized.

- Providing insight into current simulation optimization techniques by evaluating and classifying the techniques.

- Providing insight into the most current simulation output analysis techniques.

- Coupling a "Rule-based system" with the simulation optimization system to select the appropriate technique for a given problem.

- Making automatic decisions within the system, and helping the user the understand the system.

Recommendations for Further Research

As a result of the research conducted in this study, recommendations for future research can be made. Some examples of future research are:

(1) This research was limited to single response, unconstrained optimization problems. The system could be expanded to include multiple response optimization. Existing approaches for multiresponse optimization were briefly discussed in Chapter II. Since only unconstrained problems are considered in this study, techniques which consider constraints or penalty functions which modify the objective function could also be included in this research.

(2) A simulation generator for certain type of problems (e.g., queueing or inventory) could be combined with the system to assist in the development of simulation models for that type of problem.

(3) Combining graphical output with the optimization programs to plot the response surface might help the user to understand the parameter relationships. This visual aid would also help the user to pick a better starting point, which might speed up the optimization process and cut the response time.

(4) Cased-based reasoning (CBR) could be coupled with the optimization system. CBR involves the process of making decisions based on specific examples of what has occurred in the past, rather than a set of rules. Previous cases are stored for use in solving future problems. By making previous solutions available to the user, some short cuts could be made and past mistakes could also be avoided. This area of artificial intelligence has received limited attention compared to expert systems. Additional research is needed to focus on what are the appropriate techniques used to capture previous models, how well this approach works and what characteristics distinguish those models that receive frequent reuse.

(5) No effort has been made to reduce the number of factors for a given system. Factor screening methods could be employed to reduce the number of factors. Once the unimportant factors are determined, their values could be fixed for the rest of the simulation runs. This would reduce the dimensionality of the optimization problem and number of simulation runs required. This kind of factor screening would require pilot runs and would occur prior to the optimization process. It would be a responsibility of the executive controller for the system developed for this research.

BIBLIOGRAPHY

Adam, N.R., "Achieving a Confidence Interval for Parameters Estimated by Simulation," Management Science, Vol. 29, pp. 856-866, 1983.

Alexopoulos, C., "Advanced Simulation Output Analysis for a Single System," Proceedings of the 1993 Winter Simulation Conference, pp. 89-96, 1993.

Andrews, R.W. and T.J. Schreiber, "Two ARMA Based Confidence Interval Procedures for the Analysis of Simulation Output," Working Paper, Graduate School of Business Adm., University of Michigan, 1982.

Azadivar, F., "A Tutorial on Simulation Optimization," Proceedings of the 1992 Winter Simulation Conference, pp.198-204, 1992.

Azadivar, F. and J. Talavage, "Optimization of Stochastic Simulation Models," Mathematics and Computers in Simulation, Vol. 22, pp. 231-241, 1980.

Banks,J., "Selecting Simulation Software," Proceedings of the 1991 Winter Simulation Conference, pp. 15-20, 1991.

Banks, J., E. Aviles, J.R. McLaughlin, and R.C. Yuan, "The Simulator: New Member of the Simulation Family," Interfaces, Vol. 21, No. 2, pp. 76-86, 1991.

Banks, J. and J.S. Carson, Discrete-Event System Simulation, Prentice Hall, New Jersey, 1984.

Banks, J., D. Goldsman and J. S. Carson,II., Handbook of Statistical Methods for Engineers and Scientists , Chapter 12, pp. 1-36, 1990.

Barton, R. R. and J. S. Ives, Jr., "Modifications of the Nelder-Mead Simplex Method for Stochastic Simulation Response Optimization," Proceedings of the 1992 Winter Simulation Conference , pp. 945-953, 1992.

Barton, R. R., "Testing Strategeies for Simulation Optimization," Technical Report, Cornell University, 1987.

Bazaara, M.S. and C.M. Shetty, Nonlinear Programming Theory and Algorithms, John Wiley and Sons, New York, 1979.

Bengu, G., "A Simulation Optimization System with Direct Search Procedures," Doctoral dissertation, Clemson University, Clemson, SC, 1987.

Bengu, G. and J. Haddock, "A Generative Simulation Optimization System," Computers and Industrial Engineering ,Vol. 10, No. 4, pp. 301-313, 1986.

Biles, W. E. "Strategies for Optimization of Multiple- Response Simulation Models," Proceedings of the 1977 Winter Simulation Conference , pp. 135-142, 1977.

Biles, E. W. and J. J. Swain, "Optimization of Multiple- Response Simulation Models," Modeling and Simulation, Vol. 8, No. 2, pp. 991-995, 1982.

Billingsley, P., Convergence of Probability Measures, John Wiley, New York, 1968.

Birta, L. G., "A Parameter Module for CSSL-based Simulation Software," Simulation, Vol. 28, pp. 113-123, 1977.

Birta, L.G., "Optimization in Simulation Studies," Simulation and Model-Based Methodologies: An Integrative View, Edited by T. I. Oren, Springer-Verlag, Berlin, 1984.

Box, G.E. and N.R. Draper, Emprical Model-Building and Response Surfaces, John Wiley & Sons, New York, 1987.

Brightman, H.J., "Optimization Through Experimentation: Applying Response Surface Methods," Decision Sciences, Vol. 9, pp. 481-495, 1978.

Brooks, S. H., "A Comparison of Maximum Seeking Methods," Operations Research , Vol 7, No. 4, pp. 430-457, 1959.

Bulgak, A.A. and J.L. Sanders, "Integrating a Modified Simulated Annealing Algorithm with the Simulation of a Manufacturing System to Optimize Buffer Sizes in Automatic Assembly Systems," Proceedings of the 1988 Winter Simulation Conference, pp. 684-690, 1988.

Burdick, D.S. and T.H. Taylor, "Design of Computer Simulation Experiments for Industrial Systems," Comm. of the ACM, Vol. 9, pp. 329-338, 1968.

Cao, X.R., "Convergence of Parameter Sensitivity Estimates in Stochastic Experiment," IEEE Transactions on Automatic Control, Vol. 30, No. 8, pp. 834-845, 1985.

Cao, X.R., "First Order Perturbation Analysis of a Simple Multiclass Finite Source Queue," Performance Evaluation, Vol. 7, pp. 31-41, 1987.

Cao, X.R., "On a Sample Performance Function of Jackson Queueing Networks," Operations Research, Vol. 36, pp. 128-136, 1988.

Cao, X.R. and Y.C. Ho, "Perturbation Analysis of Sojourn Time in Queueing Networks," Proceedings of 22nd IEEE Conference on Decision and Control, pp. 1025-1029, 1983.

Cao, X.R. and Y.C. Ho, "Sensitivity Analysis and Optimization of Throughput in a Production Line with Blocking," IEEE Journal of Automatic Control, Vol. 32, No. 11, pp.959-967, 1987.

Carrol, C. W., "The Created Response Surface Technique for Optimizing Nonlinear, Restrained Systems," Operations Research , Vol. 9, pp. 169-184, 1961.

Cassandras, C.G., "On-line Optimization of a Flow Control Strategy," IEEE Journal of Automatic Control, Vol. 32, No. 11, pp. 1014-1017, 1987.

Charnes, J. M., "Statistical Analysis of Output Progress," Proceedings of 1993 Winter Simulation Conference, pp. 41-48, 1993.

Cherny, V. "Thermodynamical Approach to Traveling SalesmanProblem: An Efficient Simulation Algorithm," Journal of Optimization Theory Applications., Vol.45, pp. 41-45, 1985.

Clayton, E. R., W. E. Weber and B. W. Taylor,"A Goal Programming Approach to the Optimization of Multiresponse Simulation Models," IIE Transactions, Vol. 14, No. 4, pp. 282-287, 1982.

Cochran, J.K. and J. Chang, "Optimization of Multivariate Simulation Models Using a Group Screening Method," Computers and Industrial Engineering, Vol. 18, No. 1, pp. 95-103, 1990.

Conway, R.W., "Some Tactical Problems in Digital Simulation," Management Science, Vol. 10, pp. 47-61, 1963.

Cooley, B.J. and E.C. Houck, "A Variance-reduction Strategy for RSM Simulation Studies," Decision Sciences, Vol. 13, pp. 303-321, 1982.

Crane, M.A. and D.L. Iglehart, "Simulating Stable Stochastic Systems, III: Regenerative Processes and Discrete event Simulations," Operations Research, Vol 23, pp. 33-45, 1975.

Daugherty, A.F. and M.A. Turnquist, "Budget Constrained Optimization of Simulation Models via Estimation of their Response Surfaces," Operations Research, Vol. 29, pp.485-500, 1981.

Ermoliev, Y. and N.Z. Shor, "Method of Random Search for Two-Stage Problems of Stochastic Programming and its Generalization," Kibernetica, Vol.1, pp. 19-26, 1968.

Ermoliev, Y., "Stochastic Quasigradient Methods and their Application to System Optimization," Stochastics, Vol. 9, pp. 1-36, 1983.

Evtushenko, Y.P, "Numerical Methods for Finding Global Extrema of a Non-uniform Mesh," USSR Computing Machines and Mathematical Physics, Vol. 11, pp. 1390-1403, 1971.

Farrel, W., C. McCall and E. Russell, "Optimization Techniques for Computerized Simulation Models," Report to Office of Naval Research, NTIS AD-A011 844/8G1, 1975.

Farrel, W., "Literature Review and Bibliography of Simulation Optimization," Proceedings of the 1977 Winter Simulation Conference, pp.117-124, 1977.

Fegan, J.M., G.M. Lane and P.J. Nolan, "Introduction to Simulation Using Intelligent Simulation Interface (ISI)," Proceedings of the 1991 Winter Simulation Conference, pp. 143-147, 1991.

Fishman, G.S., "Estimating Sample Size in Computer Simulation," Management Science, Vol. 18, pp.21-38, 1971.

Fishman, G.S., Concepts and Methods in Discrete Event Digital Simulation, Wiley, New York, 1973.

Fishman, G.S., "Achieving Specific Accuracy in Simulation Output Analysis," Communications of ACM, Vol. 27, pp. 310-315, 1977.

Fishman, G.S., "Grouping Observations in Digital Simulation," Management Science, Vol. 24, pp. 510-521, 1978a.

Fishman, G.S., Principles of Discrete Event Simulation, Wiley, New York, 1978b.

Fox, B.L., "Implementation and Relative Efficiency of Quasirandom Sequence Generators," Working Paper, Montreal Canada, 1984.

Fu, M.C. and Y.C. Ho, "Using Perturbation Analysis for Gradient Estimation, Averaging and Updating in a Stochastic Approximation Algorithm," Proceedings of the 1988 Winter Simulation Conference, pp. 509-517, 1988.

Garcia-Diaz, A., G.L. Hogg, D.T. Phillips and E.J. Worrell, "Combined Simulation and Network Anaysis for the Production System," Simulation, Vol. 40, pp. 59-66, 1983.

Glynn, P.W., "Optimization of Stochastic Systems," Proceedings of the 1986 Winter Simulation Conference, pp. 356-365, 1986.

Goldsman, D., "Simulation Output Analysis," Proceedings of the 1992 Winter Simulation Conference, pp. 97-103, 1992.

Goldsman, D. and B.L. Nelson, "Methods for Selecting the Best System," Proceedings of the 1991 Winter Simulation Conference, pp. 177-186, 1991.

Goldsman, D. and L.W. Schruben, "Asymptotic Properties of Some Confidence Interval Estimators," Technical Report, School of Operations Research and Industrial Engineering, Cornell University, 1982.

Gong, W.B. and Y.C. Ho, "Smoothed (Conditional) Perturbation Analysis of Discrete Event Dynamic Systems," IEEE Transactions on Automatic Control, Vol. 32, No. 10, pp. 858-866, 1987.

Gottfried, B.S. and J. Weisman, Introduction to Optimization Theory , Prentice-Hall Inc., Englewood Cliffs, NJ, 1973.

Gray, H.L., G.D. Kelly, and D. McIntire, "A New Approach to ARMA Modeling," Communications in Statistics, B7, pp. 1-17, 1978.

Gross, D. and C.M. Harris, Fundamentals of Queuing Theory, Wiley, New York, 1974.

Haddock, J. and J. Mittenhal, "Simulation Optimization using Simulated Annealing," Computers & Industrial Engineering, Vol. 22, No. 4, pp. 387-395, 1992.

Haider, S. W. and J. Banks, "Simulation Software Products For Analyzing Manufacturing Systems," Industrial Engineering, Vol. 18. , No. 6, pp. 98-103, 1986.

Hajek, B., "Cooling Schedules for Optimal Annealing," Mathematics of Operations Research, Vol. 13, No. 2, pp. 311-329, 1988.

Healy, K., "Retrospective Simulation Response Optimization", Proceedings of the 1991 Winter Simulation Conference, pp. 901-906, 1991

Heidelberger, P., X.R. Cao, M. Zazanis, and R. Suri, "Convergence Properties of Infinitesimal Perturbation Analysis Estimates," Management Science, Vol. 34, No. 11, pp. 1281-1302, 1988.

Heller, N.B. and G.E. Staats, "Response Surface Optimization when Experimental Factors are Subject to Costs and Constraints," Technometrics, Vol. 15, pp. 113-123, 1973.

Hill, W.J. and W.G. Hunter, "A Review of Response Surface Methodology: a literature Survey," Technometrics, Vol. 8, No. 6, pp. 571-590, 1966.

Ho,Y.C., "On the Perturbation Analysis of Discrete-Event Dynamic Systems,"Journal of Optimization Theory and Applications, Vol. 46, No. 4, pp. 535-545, 1985.

Ho, Y.C., "Performance Evaluation and Perturbation Analysis of Discrete Event Dynamic Systems," IEEE Transactions on Automatic Control , Vol. 32, No. 7, pp. 563-572, 1987.

Ho, Y.C. and X.R. Cao, "Perturbation Analysis and Optimization of Queueing Networks," Journal of Optimization Theory and Applications, Vol. 40, No. 4, pp. 559-582, 1983.

Ho, Y.C. and X.R. Cao, "Performance Sensitivity to routing Changes in Queueing Networks and Flexible Manufacturing Systems Using Perturbations Analysis," IEEE Journal of Robotics and Automation, Vol RA-1, No. 4, pp. 165-172, 1985.

Ho, Y.C., X.R. Cao, and C. Cassandras, "Infinitesimal and Finite Perturbation Analysis for Queueing Networks," Automatica, Vol. 19, pp. 439-445, 1983.

Ho, Y.C. and C. Cassandras, "A New Approach to the Analysis of Discrete event Dynamic Systems," Automatica, Vol. 19, pp.149-167, 1983.

Ho, Y.C., M.A. Eyler, and T.T. Chien, "A Gradient Technique For General Buffer Storage Design in a Serial Production Line," International Journal of Production Research , Vol. 17, No. 6, pp. 557-580, 1979.

Ho, Y.C., M.A. Eyler, and T.T. Chien, "A New Approach to Determine Parameter Sensitivities of Transfer Lines," Management Science, Vol. 29, No. 6, pp. 700-714, 1983.

Ho, Y.C., S. Li, and P. Vakili, "On the Efficient Generation of Discrete Event Sample Paths Under Different Parameter Values," IEEE Transactions on Automatic Control, Vol. 33, No. 5, pp. 427-438, 1988.

Huq, Z., "A Process Oriented Manufacturing System Simulation to Measure the Effect of Shop Control Factors", Simulation, Vol. 62, N0. 4, pp. 218-228, 1994.

Jacobson, S.H., A.H. Russ, and L.W. Schruben, "Driving Frequency Selection for Frequency Domain Simulation Experiments," Operations Research, Vol. 39, No.6, pp. 917-924, 1991.

Jacobson, S.H. and L.W. Schruben, "Techniques For Simulation Response Optimization," Operation Research Letters, Vol. 8, pp. 1-9, 1989.

Kelton, W.D., "Statistical Analysis Methods Enhance Usefulness, Reliability Of Simulation Models," Industrial Engineering, Vol. 18, No. 9, pp. 74-84, 1989.

Khuri, A. and J. A. Cornell, Response Surfaces, Designs and Analyses, Marcel Dekker, Inc., NewYork 1987.

Kiefer, J. and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," Annals of Mathematical Statistics, Vol. 23, pp. 462-466, 1952.

King, C. U. and E.L. Fisher, "BARBS: integrating simulation, optimization and knowledge-based techniques to identify and eliminate production bottlenecks," International Journal of Computer Integrated Manufacturing, Vol. 2, No. 6, pp. 317-328, 1989.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, No. 4598, pp. 671-680, 1983.

Kleijnen, J.P.C., Statistical Techniques in Simulation, Part I, Marcel Dekker, New York, 1974.

Kleijnen, J.P.C., Statistical Techniques in Simulation, Part II, Marcel Dekker, New York, 1975.

Kumar, P.R., "On Simulating to Estimate Derivatives of Performance Measures in Discrete Event Systems," Proceedings of the 23rd Conference on Decision and Control, pp. 534, 1984.

Kuester, J.L. and J.H. Mize, Optimization techniques with FORTRAN, Mc Graw-Hill Inc., NewYork, 1973.

L'Ecuyer, P., "An Overview of Derivative Estimation, "Proceedings of the 1991 Winter Simulation Conference, pp.207-217, 1991.

Lavenberg, S.S. and C.H. Sauer, "Sequential Stopping Rules for the Regenerative Method of Simulation," IBM Journal of Res. Develop., Vol. 21, pp. 545-558, 1977.

Law, A.M., "Confidence Intervals in Discrete Event Simulation: A Comparison of Replication and Batch Means," Naval Research Logistics Quarterly, Vol. 24, pp. 667-678, 1977.

Law, A.M., "Statistical Analysis of Simulation Output Data," Operations Research, Vol.31, No.6, pp.983-1029, 1983.

Law, A.M., "Simulation Software for Manufacturing Applications," Industrial Engineering, Vol. 22, No. 7, pp. 18, 1990.

Law, A.M. and J.S. Carson, "A Sequential Procedure for Determining the Length of a Steady State Simulation," Operations Research, Vol. 27, pp. 1011-1025, 1979.

Law, A.M. and S. W. Haider, "Selecting Simulation Software for Manufacturing Applications: Practical Guidelines & Software Survey," Industrial Engineering, Vol. 21, No. 5, pp. 33-46, 1989.

Law, A.M. and W.D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Publishing Co., New York, 1982a.

Law, A.M. and W.D. Kelton, "Confidence Intervals for Steady-State Simulations, II: A Survey of Sequential Procedures," Management Science, Vol. 28, No. 5, pp. 550-562, 1982b.

Law, A.M. and W.D. Kelton, "Confidence Intervals for Steady-State Simulations, I: A Survey of Fixed Sample Size Procedures," Operations Research, Vol 32, No. 6, pp. 1221-1239, 1984.

Law, A.M. and W.D. Kelton, Simulation Modeling and Analysis, 2nd ed., McGraw-Hill Publishing Co., New York, 1991.

Law, A.M. and M.G. McComas, "Secrets of Successful Simulation Studies," Industrial Engineering, Vol. 22, No. 5, pp. 47-53, 1990.

Lee, Y. and K. Iwata, "Part Ordering Through Simulation Optimization in An FMS," International Journal of Production Research, Vol. 29, No. 3, pp. 1309-1322, 1991.

Liu, C.M. and J.L. Sanders, "Stochastic Design Optimization of Asynchronous Flexible Assembly Systems," Annals of Operations Research, Vol. 15, pp. 131-154, 1988.

Ma, X. and A. K. Kochhar, "A Comparison Study of Two Tests for Detecting Initialization Bias in Simulation Output," Simulation, Vol. 61, No. 2, pp. 94-101, 1993.

MacNair, E. A., "Toward a Higher Level, Output Analysis Interface", Proceedings of the 1991 Winter Simulation Conference, pp. 822-831, 1991.

Manz, E.M., J. Haddock and J. Mittenthal, "Optimization of an Automated Manufacturing System Simulation Model Using Simulated Annealing," Proceedings of the 1989 Winter Simulation Conference, pp. 390-394, 1989.

McBride, R. D. and D. E. O'leary, "The use of mathematical programming with artificial intelligenceand expert systems," European Journal of Operational Research, Vol. 70, pp. 1-15, 1993.

Mechanic, H. and W. McKay, "Confidence Intervals for Averages of Dependent Data in Simulations II," Technical Report No. ASDD 17-202, IBM Corporation, Yorktown Heights, New York, 1966.

Meidt, G. J. and K. W. Bauer,Jr., "PCRSM: A Decision Support System for Simulation Metamodel Construction", Simulation, Vol. 59, No. 3, pp183-191, 1992.

Meketon, M.S., "Optimization in Simulation: A Survey of Recent Results," Proceedings of the 1987 Winter Simulation Conference, pp. 58-67, 1987.

Metropolis, N., N. Rosenbluth, A. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," Journal of Chemical Physics, Vol. 21, pp. 1087-1092, 1953.

Mihram, G.A., "An Efficient Procedure for Locating the Optimal Simulation Response," Proceedings of the 4th Conference on the Application of Simulation, pp. 154-161, 1970.

Miller, R. G., "The Jackknife - A Review," Biometrika, Vol. 61, pp. 1-15, 1974.

Mitra M. and S. K. Park, "Solution to the Indexing Problem of Frequency Domain Simulation Experiments, "Proceedings of the 1991 Winter Simulation Conference , pp. 907-920, 1991.

Montgomery, D.C., "Methods for Factor Screening in Computer Simulation Experiments," Technical Report, Georgia Institute of Technology, 1979.

Montgomery, D. C. and V. M. Bettencourt, "Multiple Response Surface Methods in Computer Simulation," Simulation ,Vol. 29, No. 4, pp. 113-121, 1977.

Montgomery, D.C. and D.M. Evans, Jr., "Second Order Response Surface Designs in Computer Simulation," Simulation, Vol. 25, pp. 169-178, 1975.

Moore, R. D. and B. J. Lee, "Simulation for Closed Loop Factory Optimization," Proceedings of the 1989 Winter Simulation Conference, pp. 237-242, 1989.

Myers, R.H., Response Surface Methodology, Allyn-Bacon, Boston, 1971.

Nandkeolyar, U. and D.P. Christy, "Using Computer Simulation to Optimize Flexible Manufacturing System Design," Proceedings of the 1989 Winter Simulation Conference, pp. 396-405, 1989.

Nelder, J. A. and R. Mead, "A Simplex Method for Function Minimization," The Computer Journal, Vol. 7, pp. 308-313, 1965.

Nozari, A., S.F. Arnold, and C.D. Pegden, "Statistical Analysis for Use with the Schruben and Margolin Correlation Induction Strategy," Operations Research, Vol. 35, pp. 127-139, 1987.

Olsson, D. M., "A Sequential Simplex Program For Solving Minimization Problems," Journal of Quality Technology , Vol. 6, No. 1, pp. 53-57, 1974.

Olsson, D.M. and L.S. Nelson, "The Nelder-Mead Simplex Procedure for Function Minimization," Technometrics, Vol. 17, No. 1, pp. 45-51, 1975.

Pegden, C.D. and M.P. Gately, "A Decision-Optimization Module for SLAM," Simulation, Vol. 34, pp. 18-25, 1980.

Pflug, G.C., "Optimizing Simulated System," Simuletter, Vol. 15, No. 4, pp. 6-9,1984.

Plackett, R. L. and J. P. Burman, "The Design of Multifactor Experiments," Biometrika, Vol. 33, pp. 305-325, 1946.

Pritsker, A.A.B., Introduction to Simulation and SLAM II, System Publishing Co., Indiana, 1986.

Pritsker, A.A.B., C. E. Sigal, and R. D. J. Hammesfahr, SLAM II Network Models for Decision Support, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1989.

Ramachandran, V., D.L. Kimbler, and G. Naadimuthu, "Expert Post-Processor for Simualtion Output Analysis", Computers and Industrial Engineering, Vol. 15, No. 1-4, pp. 98-103, 1988.

Rees, L. P., E. C. Clayton, and B. W. Taylor, "Solving Multiple Response Simulation Models Using Modified Response Surface Methodology Within A Lexicographic Goal Programming Framework," IIE Transactions , Vol. 17, No. 1,pp. 47-57, 1985.

Robbins, H. and S. Monro, "A Stochastic Approximation Method ," Annals of Mathematical Statistics, Vol. 22, pp. 400-407, 1951.

Rooks, M., "A Unified Framework for Visual Interactive Simulation," Proceedings of the 1991 Winter Simulation Conference, pp. 1146-1155, 1991.

Rooks, M., "A User-Centered Paradigm for Interactive Simulation," Simulation, Vol. 60, No. 3, pp. 168-177, 1993.

Ruppert, D., R.L. Reish, R.B. Deriso and R.J. Carroll, "Optimization Using Stochastic Approximation and Monte Carlo Simulation (with application to harvesting atlantic menhaden)," Biometrics, Vol.40, pp.535-545, 1984.

Rustagi, J.S., "Optimization in Simulation," Technical Report, Department of Statistics, Ohio State University, 1981.

Safizadeh, M. H., "Optimization in Simulation: Current Issues and Future Outlook," Naval Research Logistics, Vol. 37, pp. 807-825, 1990.

Safizadeh, M.H. and B.M. Thornton, "Optimization in Simulation Experiments Using Response Surface Methodology," Computers and Industrial Engineering, Vol. 8, No. 1, pp. 11-27, 1984.

Sargent, R., "Research Issues in Metamodeling", Proceedings of the 1991 Winter Simulation Conference, pp. 888-893, 1991.

Sargent, R., K. Kang and D. Goldsman, "An Investigation of Finite-Sample Behavior of Confidence Interval Estimators", Operations Research, Vol. 40, No. 5, pp. 898-913, 1992.

Schittkowski, K., Nonlinear Programming Codes, Springer-Verlag, Berlin, Heidelberg, New York 1980.

Schmeiser, B.W., "Batch Size Effects in the Analysis of Simulation Output," Operations Research, Vol. 30, pp. 556-568, 1982.

Schmeiser, B.W. and K. Kang, "Properties of Batch Means from Stationary ARMA(1,1) Time Series," Technical Report 81-3, School of Industrial Engineering, Purdue University , 1981.

Schreiber, T.J. and R.W. Andrews, "Interactive Analysis of Simulation Output by the Method of Batch Means," Proceedings of the 1979 Winter Simulation Conference, pp. 512-524, 1979.

Schruben, L.W., "Control of Initialization Bias in Multivariate Simulation Response," Communications of ACM, Vol. 24, pp.246-252, 1981.

Schruben, L.W., "Detecting Initialization Bias in Simulation Output," Operations Research, Vol.30, pp 569-590, 1982.

Schruben, L.W., "Confidence Interval Estimation Using Standardized Time Series," Operations Research, Vol. 31, pp. 1090-1118, 1983.

Schruben, L.W., "Simulation Optimization Using Frequency Domain Methods," Proceedings of the 1986 Winter Simulation Conference, pp. 366-369, 1986.

Schruben, L.W. and V.J. Cogliano, "An Experimental Procedure for Simulation Response Surface Model Identification," Communications of the ACM, Vol. 30, No. 8, pp.716-730, 1987.

Schruben, L.W. and B.H. Margolin, "Pseudorandom Number Assignment in Statistically Designed Simulation and Distribution Sampling Experiments," Journal of the American Statistical Association, Vol. 73, pp. 503-520, 1978.

Schruben, L.W., H. Sing, and L. Tierney, "Optimal Tests for Initialization Bias in Simulation Output," Operations Research, Vol. 31, No. 6, pp. 1167-1178, 1983.

Seila, A.F., "Output Analysis for Simulation," Proceedings of the 1990 Winter Simulation Conference, pp. 49-54, 1990.

Seila, A.F., "Advanced Output Analysis," Proceedings of the 1992 Winter Simulation Conference, pp. 190-197, 1992.

Seila, A.F. and J. Banks., "Spreadsheet Risk Analysis Using Simulation," Simulation, Vol. 55, pp. 163-170, 1990.

Smith, D.E., "Requirements of an Optimizer For Computer Simulation," Naval Research Logistics Quarterly, Vol. 20, pp. 161-179, 1973a.

Smith, D.E., "An Empirical Investigation of Optimum Seeking in the Computer Simulation Situation," Operation Research, Vol. 21, pp. 475-497, 1973b.

Smith, D.E., "Automatic Optimum -seeking Program for Digital Simulation," Simulation, Vol. 27, pp.27-32, 1976.

Smith, D.E. and C.A. Mauro, "Factor Screening in Computer Simulation," Simulation, Vol. 38, pp. 49-54, 1982.

Spendley, W., G. R. Hext, and F.R. Himsworth, "Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation," Technometrics, Vol. 4, pp. 441-461, 1962.

Standridge, C. R. and J. Tsai, "A Method for Computing Discrete Event Simulation Performance Measures from Traces", Simulation, Vol. 56, No. 6, pp. 384-391, 1992.

Starr, N., "The Performance of a Sequential Procedure for the Fixed-Width Interval Estimation of the Mean," Annals of Mathematical Statistics ,Vol. 37, pp. 36-50, 1966.

Stuckman, B., G. Evans and M. Molloghasemi, "Comparison of Global Search Methods for Design Optimization Using Simulation," Proceedings of the 1991 Winter Simulation Conference, pp. 937-944, 1991.

Suri, R.,"Implementation of Sensitivity Calculations on a Monte Carlo Experiment," Journal of Optimization Theory and Applications, Vol. 40., No. 4, pp. 625-630, 1983a.

Suri, R., "Infinitesimal Perturbation Analysis of Discrete Event Dynamic Systems: A General Theory," Proceedings of the 22nd Conference on Decision and Control, pp. 1030-1038, 1983b.

Suri, R., "Infinitesimal Perturbation Analysis for General Discrete Event Systems," Journal of ACM, Vol. 34, No. 3, pp. 686-717, 1987.

Suri, R., "Perturbation Analysis: The State of the Art and Research Issues Explained via the GI/GI/1 Queue," Proceedings of the IEEE, Vol. 77, No. 1, pp. 114-137, 1989.

Suri, R. and J. W. Dille, "A Technique for On-line Sensitivity Analysis of Flexible Manufacturing Systems," Annals of Operations Research , Vol. 3, pp. 381-391, 1985.

Suri, R. and Y.T. Leung, "Single Run Optimization of a SIMAN Model for Closed Loop Flexible Assembly Systems," Proceedings of the 1987 Winter Simulation Conference, pp. 738-748, 1987.

Suri, R. and Y.T. Leung, "Single Run Optimization of Discrete Event Simulations - An Empirical Study Using the M/M/1 Queue." IEEE Transactions, Vol. 21, No. 1, pp. 35-48, 1989.

Suri, R. and M. Zazanis, "Perturbation Analysis Gives Strongly Consistent Sensitivity Estimates for the M/G/1 Queue," Management Science, Vol. 34, No. 1, pp. 39-64, 1988.

Taha, H. A., Simulation Modeling and Simnet, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1988.

Taha, H. A., Operations Research, MacMillan Publishing Co., 4th ed., New York, 1987.

Tew, J.D. and J.R. Wilson, "Metamodel Estimation Using Integrated Correlation Methods," Proceedings of the 1987 Winter Simulation Conference, pp. 409-413, 1987.

Vecchi, M.P. and S. Kirkpatrick, "Global Wiring by Simulated Annealing," IEEE Trans. Computer-Aided Design, Vol. 2, No. 4, pp.215-222, 1983.

Wilde, D.J. and C. S. Beightler, Foundations of Optimization, Prentice Hall, New Jersey 1967.

Wilhelm, M.R. and T.L. Ward, "Solving Quadratic Assignment Problems by Simulated Annealing," IIE Transactions, Vol. 19, No. 1, pp. 107-119, 1987.

Wilson, J.R., "Future Directions in Response Surface Methodology," Proceedings of the 1987 Winter Simulation Conference, pp. 378-381, 1987.

Wilson, J.R. and A.A.B. Pritsker, "A Survey of Research on the Simulation Startup Problem," Simulation, Vol. 31, pp. 55-58, 1978a.

Wilson, J.R. and A.A.B. Pritsker, "A Procedure for Evaluating Startup Policies in Simulation Experiments," Simulation, Vol. 31, pp. 79-89, 1978b.

Zazanis, M.A. and R. Suri, "Perturbation Analysis of GI/G/1 Queue," Queueing Systems: Theory and Applications , submitted, 1986.

Zeleny, M., Multiple Criteria Decision Making ,McGraw-Hill, New York, 1982.

Zheng, Q., "Theory and Methods for Global Optimization - An Integral Approach," presented at Optimization Days Meeting, Montreal, Canada, 1986.

Zoutendijk, G., Methods of Feasible Directions, Elsevier Science Publishers, Amsterdam, 1960.

APPENDIXES

APPENDIX A

DETAILS OF THE PHASES OF RSM

DETAILS OF THE PHASES OF RSM

First-order Design Phase and Least Squares Estimates

The true response surface may be written as (Myers (1971))

$$y = \beta_0 + \sum_{1}^{k} \beta_i x_i + \sum_{1}^{k} \sum_{1}^{k} \beta_{ij} x_i x_j + \ldots$$

An estimate of the response at $(x_1,..., x_k)$ is given by

$$\hat{y} = b_0 + \sum_{1}^{k} b_i x_i \ .$$

Denote $N = 2^{k-p}$ points in the 2^{k-p} fractional factorial by $\mathbf{x}_1,..., \mathbf{x}_N$ where

$$\mathbf{x}_j = (x_{j1},..., x_{jk})$$

and p is the largest integer such that the number of design points (2^{k-p}) is greater than k, the number of factors (except when k=3).

If y_j denotes the average observed response of the m iterations of a simulation run corresponding to point $\mathbf{x}_j$ then the estimate b_i of β_i may be obtained from the least squares equation

$$\mathbf{y} = \mathbf{x}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \ .$$

Given the matrix **x**, a function of preselected x levels, and the vector **y** of responses, the least squares method uses as an estimate of $\boldsymbol{\beta}$, that vector which results in a minimum value for the sum of squares of the errors.

$$L = \sum_{i=1}^{n} \varepsilon_i^2$$

L can be written

$$L = (y - x\hat{\beta})'(y - x\hat{\beta})$$

Upon expanding the right-hand side of the above equation

$$L = y'y - 2\beta'x'y + \beta'x'x\beta$$

$$\frac{\partial L}{\partial \hat{\beta}} = -2x'y + 2(x'x)\hat{\beta}$$

Setting the partial derivative to zero and solving for β, we have the following least squares estimator

$$\mathbf{b} = \hat{\beta} = (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y}$$

where

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ . \\ . \\ b_k \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ y_k \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} 1 & x_0' \\ 1 & x_1' \\ . & . \\ . & . \\ 1 & x_N' \end{bmatrix}$$

Because of the pattern of + 1's in the matrix **x** the b_i's are given by

$$b_o = \frac{\sum_{0}^{N} y_j}{N+1}$$

$$b_i = \frac{\sum_{1}^{N} y_j x_{ji}}{N}$$

Before the steepest ascent path is followed, the adequacy of the fit must be checked. How well the first-order equation fits the actual response surface is measured by the lack of fit test. F test is used for the lack of fit. For testing the lack of fit, the appropriate test statistics is

$$F^* = \frac{MSLF}{MSPE}$$

where MSLF is the lack of fit mean square and MSPE the pure error mean square.

Second-order Design Phase:

A second order approximation can be written as

$$\hat{y} = b_0 + \sum_{1}^{k} b_i x_i + \sum_{1}^{k}\sum_{1}^{k} b_{ij} x_i x_j$$

This equation can be written in matrix notation as

$$\mathbf{y} = b_0 + \mathbf{x}'\mathbf{b} + \mathbf{x}'\mathbf{B}\mathbf{x}$$

where

$$b = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_k \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_k \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 2b_{11} & b_{12} & . & . & b_{1k} \\ b_{21} & 2b_{22} & . & . & b_{2k} \\ . & . & . & . & . \\ . & . & . & . & . \\ b_{k1} & b_{k2} & . & . & 2b_{kk} \end{bmatrix}$$

The stationary point for the response function is found by solving

$$\frac{\partial \hat{y}}{\partial x} = \mathbf{b} + 2\mathbf{B}\mathbf{x} = 0$$

from which it follows that the stationary point $\mathbf{x}_s$ is given by

$$\mathbf{x}_S = -\mathbf{B}^{-1}\mathbf{b}/2.$$

The predicted response at this point is

$$y_S = b_0 + x_S b/2$$

Analysis of the stationary point:

The predicted response at any point can be expressed in canonical form (Myers (1971)) as

$$\hat{y} = \hat{y}_S + \sum_{1}^{k} \lambda_i \omega_i^2$$

where y_s is the estimated response at the stationary point x_s, λ_i's are the eigenvalues of matrix **B** and ω_i 's are the linear combination of the x's. This reduction of the response surface to canonical form is called canonical analysis. The nature of the stationary point and the response surface can be determined by observing the sign and magnitude of the λ_i's.

i) If all of the λ_i's are positive, x_s defines the point of predicted minimum response.

ii) If all of the λ_i's are negative, the stationary point represents a point of maximum response.

iii) If the λ_i's differ in sign, the stationary point is a saddle point.

APPENDIX B

NELDER AND MEAD SIMPLEX METHOD

NELDER AND MEAD SIMPLEX METHOD

1. Construction of the initial simplex: a starting simplex is constructed consisting of an initial point x_1 and the following additional points (Kuester & Mize (1973)):

$$\mathbf{x}_j = \mathbf{x}_1 + \boldsymbol{\xi}_j$$

where ξ_j is determined from the following table.

j	$\xi_{1,j}$	$\xi_{2,j}$	....	$\xi_{N-1,j}$	$\xi_{N,j}$
2	p	q	....	q	q
3	q	p	....	q	q
.	.	.		.	.
N	q	q	....	p	q
N+1	q	q	....	q	p

N : numberof variables a : side length of simplex

$$p = \frac{a}{N\sqrt{2}}(\sqrt{N+1} + N - 1)$$

$$q = \frac{a}{N\sqrt{2}}(\sqrt{N+1} - 1)$$

2. Location of a reflected point:

$$x_{i,j}(\text{reflected}) = x_{i,c} + \alpha(x_{i,c} - x_{i,j}(\text{worst})) \qquad i = 1,2,..., N$$

$$x_{i,c}(\text{centroid}) = \frac{1}{N}\left[\sum_{j=1}^{N+1} x_{i,j} - x_{i,j}(\text{worst})\right]$$

3. Location of a contracted point:

$$x_{i,j}(\text{contracted}) = x_{i,c} - \beta(x_{i,c} - x_{i,j}(\text{reflected})) \qquad i = 1,2,..., N$$

4. Location of an expansion point:

$$x_{i,j}(\text{expansion}) = x_{i,c} + \gamma(x_{i,j}(\text{reflected}) - x_{i,c}) \qquad i = 1,2,..., N$$

APPENDIX C

LAW AND CARSON'S BATCH MEANS METHOD

LAW and CARSON'S SEQUENTIAL BATCH MEANS METHOD:

The usual estimator of $\rho_1(l)$ is given by

$$\rho(n,k) = \frac{\sum_{j=1}^{n-1}[\bar{Y}_j(k) - Y(n,k)][\bar{Y}_{j+1}(k) - Y(n,k)]}{\sum_{j=1}^{n}[\bar{Y}_j(k) - \bar{Y}(n,k)]^2}$$

Let $Y_j(k)$ (j=1,2,...,n) be the sample (batch) mean of the k observations in the jth batch and let

$$\bar{Y}(n,k) = \frac{\sum_{j=1}^{n} Y_j(k)}{n} = \frac{\sum_{i=1}^{m} Y_i}{m}$$

be the grand sample mean.

However, $\rho_1(l)$ can be estimated by the jackknife estimator $\rho_j(n,l)$

$$\rho_j(n,k) = 2\rho(n,k) - \frac{\rho^1(n/2,k) + \rho^2(n/2,k)}{2}$$

where ρ^1 and ρ^2 are, respectively, the usual lag 1 estimators based on the first n/2 and the last n/2 batches. The steps of the algorithm are given below:

Step 1: Fix n=40, f=10, m_0=600 , m_1=800, the stopping value u=0.4, and the relative precision γ >0; let i=1, collect m_i observations.

Step 2a: Divide m_i observations into fn batches of size k = m_i /fn. Compute ρ_j(fn,k) from $Y_j(k)$ (j=1,2,...,fn). If ρ_j(fn,k) > u, go to step 3.

If ρ_j(fn,k) < 0, go to step 2c. Otherwise go to step 2b.

Step 2b: Divide m_i into fn/2 batches of size 2k. Compute $\rho_j(fn/2,2k)$ from $Y_j(2k)$ (j=1,2,...,fn/2). If $\rho_j(fn/2,2k) < \rho_j(fn,k)$ go to step2c. Otherwise, go to step 3.

Step 2c: Divide m_i into n batches of size fk. Compute Y(n,fk) and

$\delta = t_{n-1,1-\alpha/2}\sqrt{\hat{\alpha}^2[Y(n,fk)]}$ from $Y_j(fk)$ (j=1,2...,n)

If $\delta / Y(n,fk) < \gamma$, use $Y(n,fk) + \delta$ as an approximate $100(1-\alpha)$ percent confidence interval for mean ν and stop. Otherwise, go to step 3.

Step 3: Replace i by i+1, set $m_i = m_{i-2}$, collect the additional required observations r and go to step 2a.

APPENDIX D

SAMPLE MENUS

```
**********************************************************
* AUTOMATED SIMULATION OPTIMIZATION SYSTEM *
**********************************************************

> **ENTER NUMBER OF VARIABLES
> 2
> **ENTER TYPE OF OPTIMIZATION
        1- MINIMIZATION
        0-MAXIMIZATION
> 0


**********************************************************
*                  AVAILABLE METHODS                     *
**********************************************************
* 1- RESPONSE SURFACE METHODOLOGY                        *
* 2- NELDER AND MEAD SIMPLEX METHOD                      *
* 3- HOOKE AND JEEVES PATTERN SEARCH                     *
**********************************************************
> **ENTER  1- AUTOMATIC OPTIMIZATION
             2- USER SELECTED METHOD
> 2
> **ENTER SELECTED METHOD NUMBER
> 2
> **ENTER INPUT VARIABLE (1)  250
> **ENTER INPUT VARAIABLE (2)  400
```

Figure 21. Screen Display for Input Data Entry

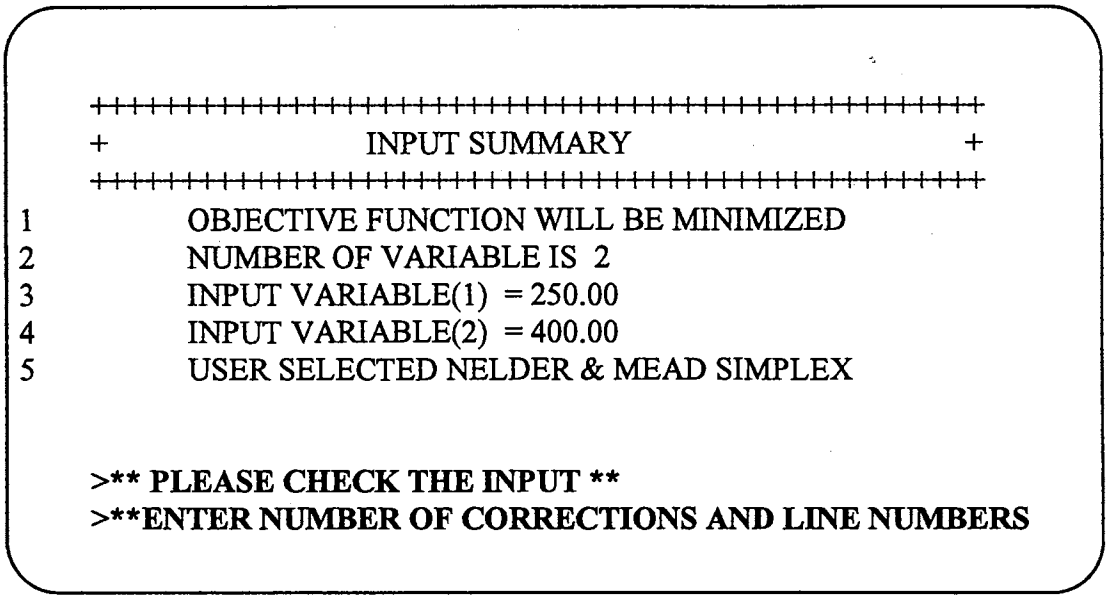


Figure 22. Screen Display for Input Verification

APPENDIX E

QUEUEING COST MODELS AND SLAM NETWORK MODELS

Introduction

This appendix presents the system performance of the (M/M/1):(GD/∞/∞) queueing model which was chosen as the first test problem. Figures 23 and 24 show the plots of time in the system and cumulative average of time in the system. Figure 25 shows the total cost as a function of the service level. The minimum total cost is $3.803 with the service level of 43.383.

The Slam network models and statements for both the (M/M/1):(GD/∞/∞) and the (M/M/1):(GD/N/∞) queueing models are given in this appendix.

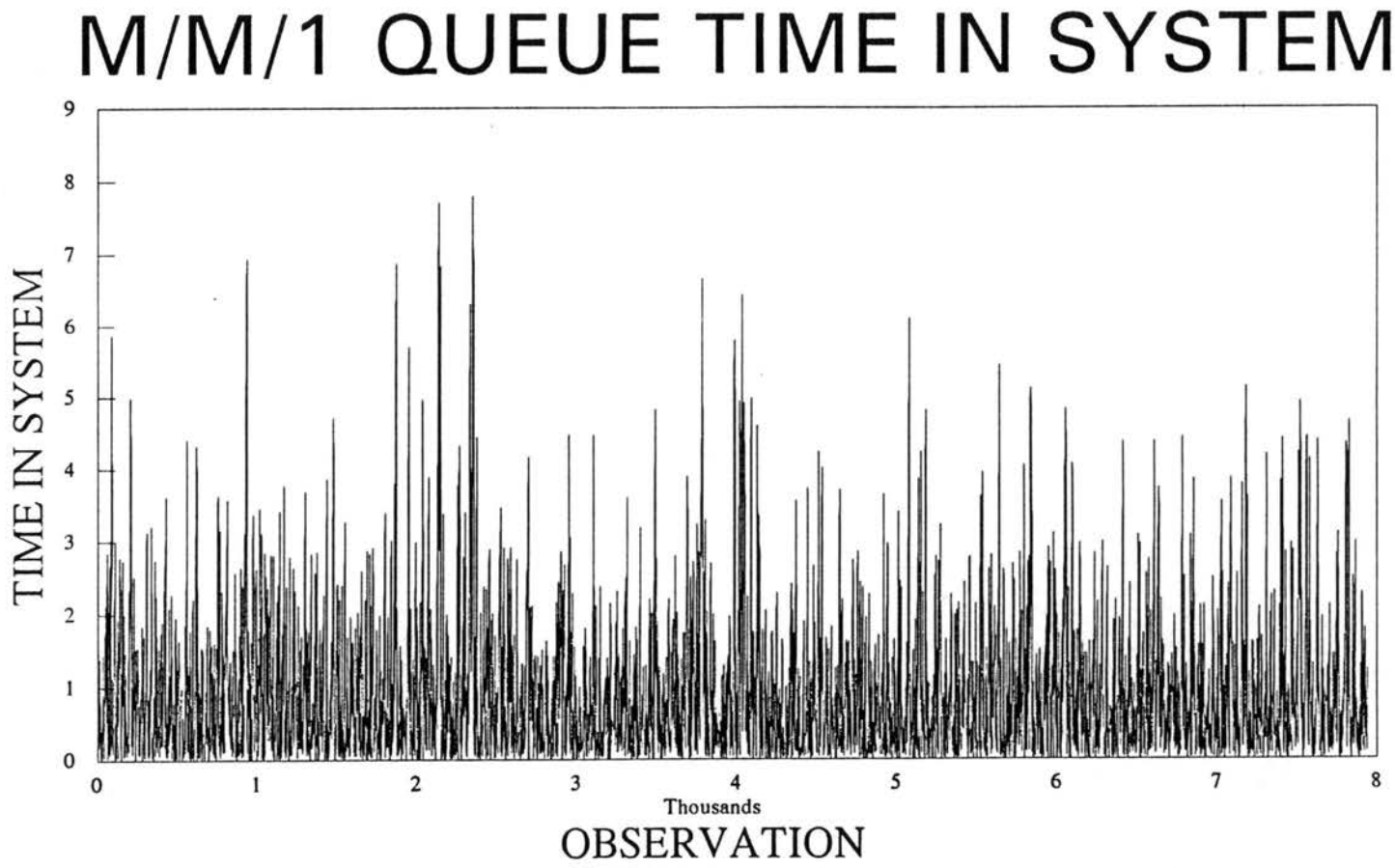


Figure 23. Time in System for M/M/1 Queue

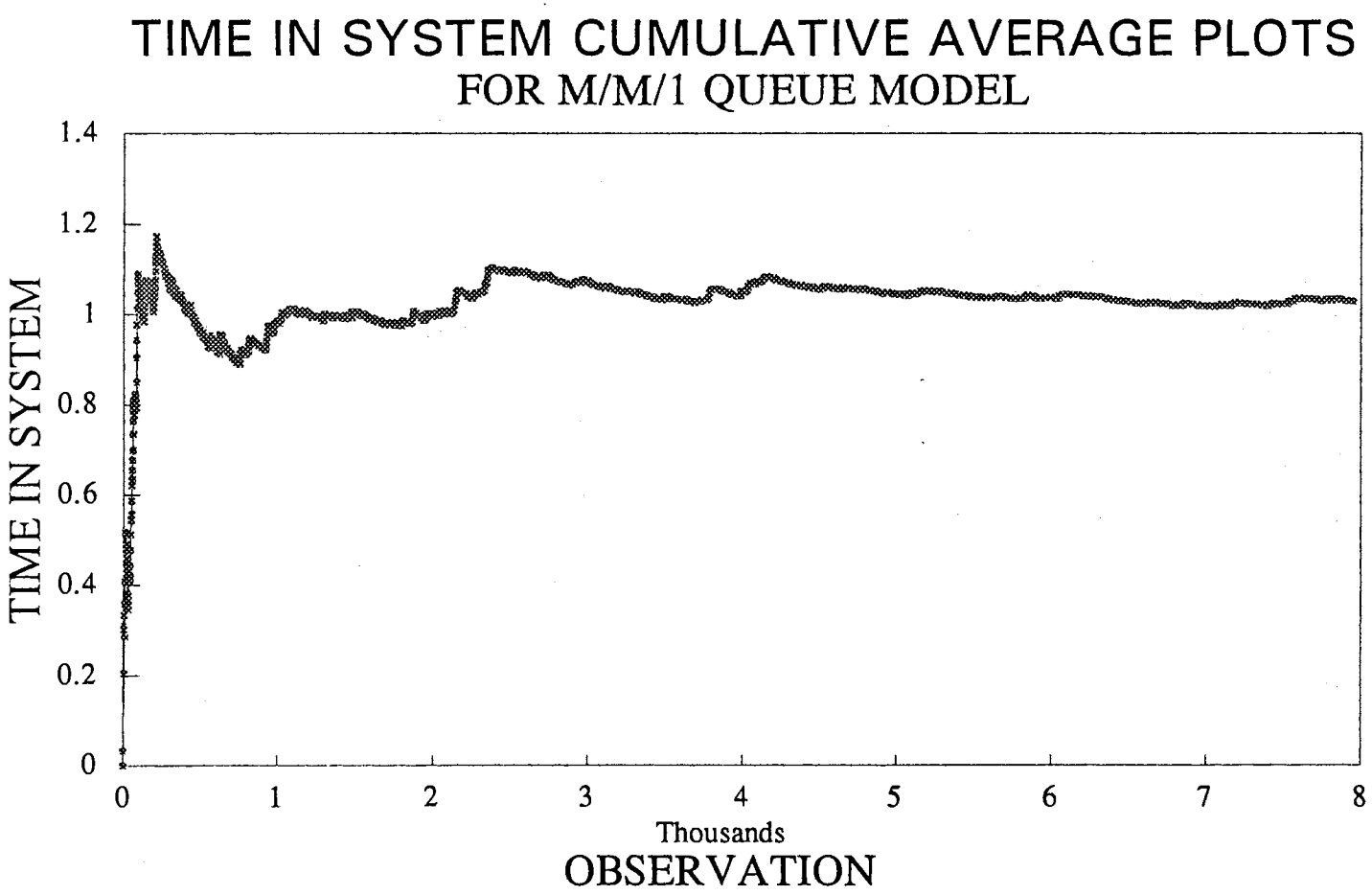


Figure 24. Time in System Cumulative Average Plots for M/M/1 Queue

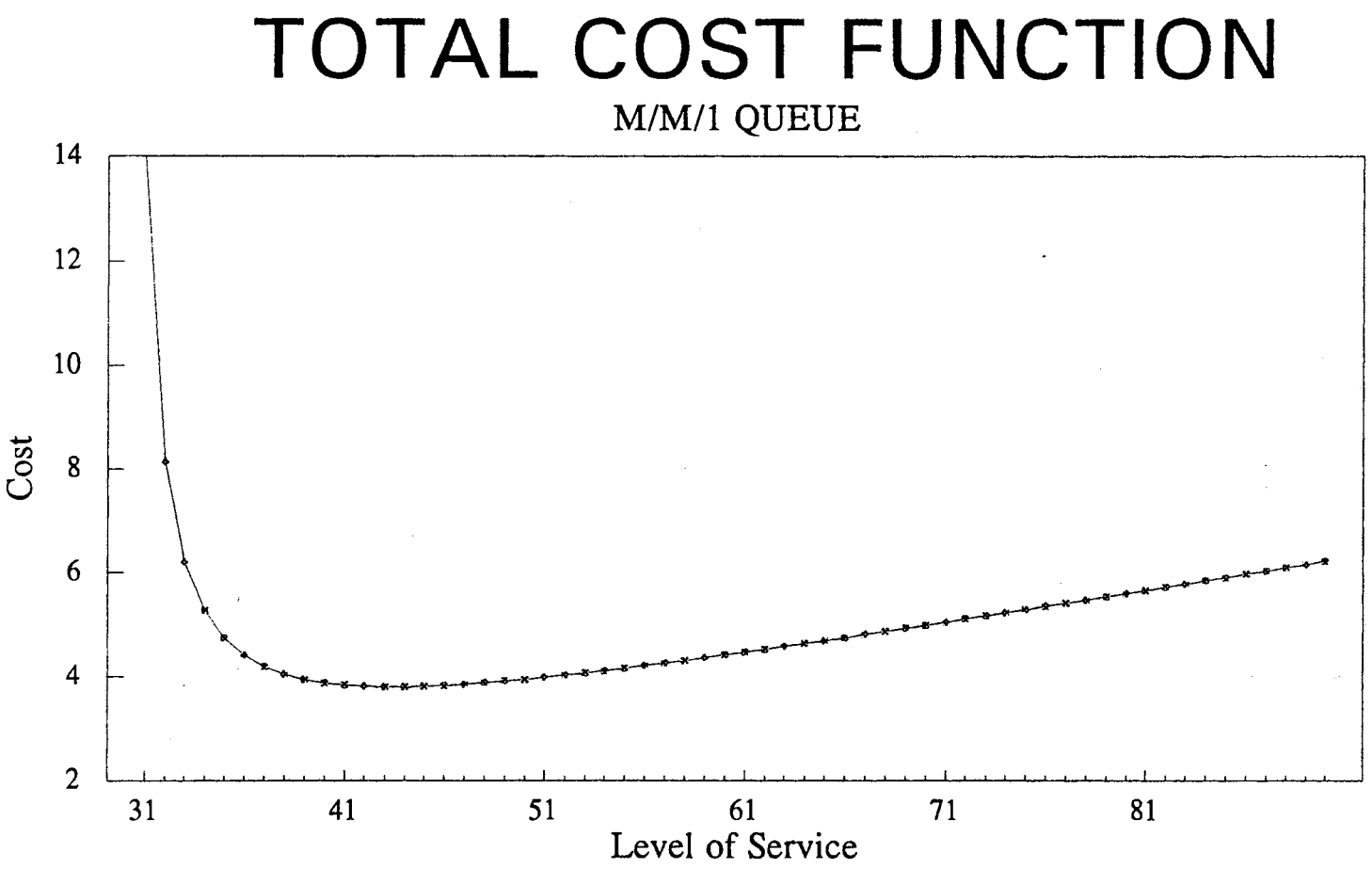
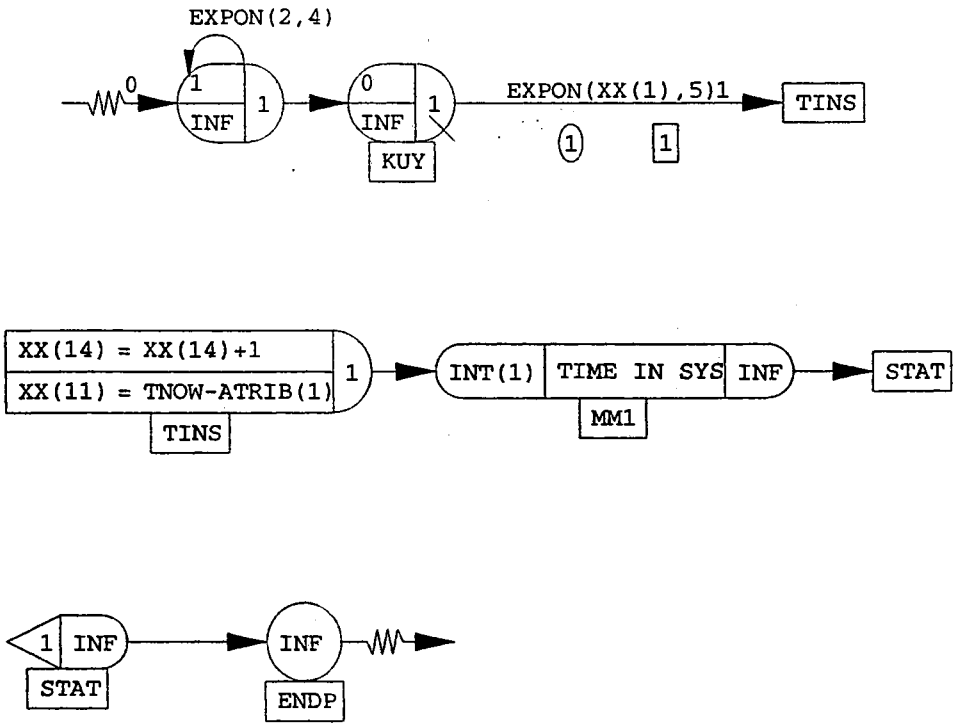
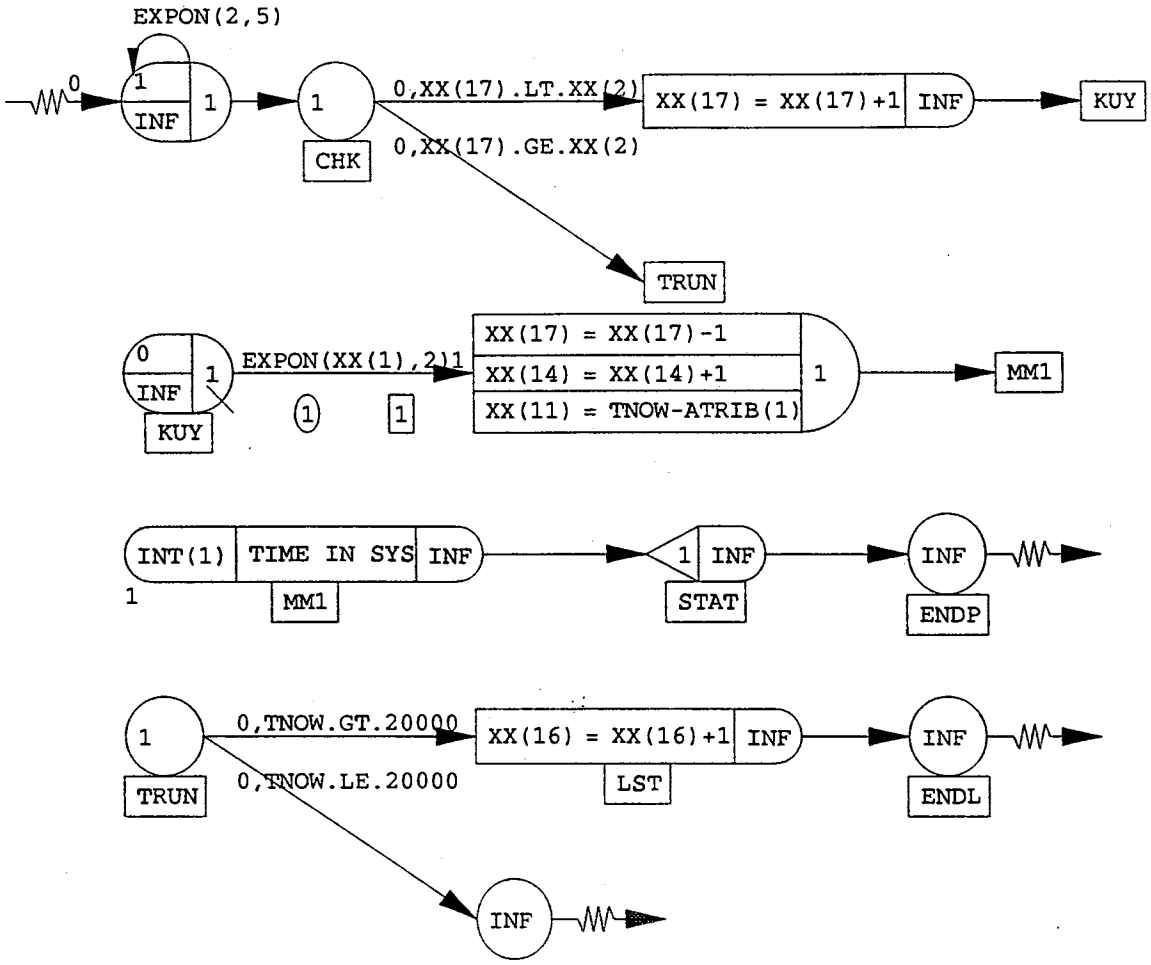


Figure 25. Total Cost Function for M/M/1 Queue



SLAM NETWORK FOR M/M/1 QUEUE

```
;*************************************************************************
;*                                                                       *
;*   SLAMSYSTEM NETWORK MODEL FOR M/M/1 QUEUEING EXAMPLE                 *
;*                                                                       *
;*************************************************************************
;       XX(1) : input variable for mean service time
;
        CREATE,EXPON(2),,1,,1;
        ACTIVITY;
KUY     QUEUE(1),,,;
        ACTIVITY(1)/1,EXPON(XX(1)),,TINS;
;
;       Calculate Time in system and number of observations
;
TINS    ASSIGN,XX(14)=XX(14)+1,XX(11)=TNOW-ATRIB(1),1;
        ACTIVITY;
MM1     COLCT,INT(1),TIME IN SYS;
        ACTIVITY,,,STAT;
;
STAT    EVENT,1;
        ACTIVITY;
ENDP    TERMINATE;
        END;
```



SLAM NETWORK FOR M/M/1 QUEUE WITH FINITE CAPACITY

```
;*************************************************************************
;*   SLAMSYSTEM NETWORK MODEL FOR (M/M/1):(GD/N/∞ ) EXAMPLE      *
;*************************************************************************
    CREATE,EXPON(2,5),,1,,1;
    ACTIVITY;
CHK  GOON,1;
    ACTIVITY,,XX(17).LT.XX(2);
    ACTIVITY,,XX(17).GE.XX(2),TRUN;
    ASSIGN,XX(17)=XX(17)+1;
    ACTIVITY,,,KUY;
;
KUY  QUEUE(1),,,;
    ACTIVITY(1)/1,EXPON(XX(1),2);
    ASSIGN,XX(17)=XX(17)-1,XX(14)=XX(14)+1,XX(11)=TNOW-ATRIB(1),1;
    ACTIVITY,,,MM1;
;
MM1  COLCT(1),INT(1),TIME IN SYS;
    ACTIVITY;
STAT EVENT,1;
    ACTIVITY,,,ENDP;
ENDP TERMINATE;
;
TRUN GOON,1;
    ACTIVITY,,TNOW.GT.20000;
    ACTIVITY,,TNOW.LE.20000,ZAAB;
LST  ASSIGN,XX(16)=XX(16)+1;
    ACTIVITY;
ENDL TERMINATE;
ZAAB TERMINATE;
    END;
```

APPENDIX F

CONTINUOUS REVIEW INVENTORY SYSTEMS

CONTINUOUS REVIEW INVENTORY SYSTEMS

A single-item continuous review inventory system was chosen as an illustrative example. The inventory position was reviewed at each demand occurrence. Whenever the inventory position dropped to or below a specified reorder point, a fixed amount of inventory (reorder quantity) was ordered. The demand quantity and the lead time to receive an ordered quantity were random. Any unsatisfied demand was considered lost. Figure 26 shows the behavior of the general continuous review inventory system. The objective was to maximize the net profit and to find optimal levels for the reorder quantity and the reorder point. The costs included a holding cost, a stock-out cost and the cost of management procedures such as inventory review and ordering costs. By combining costs and revenues, the objective function, profit, for an inventory policy for T time units can be defined as (Pritsker, Sigal and Hammesfahr (1989))

$$P = PPU*TS - CPU*TO - HC*ASV*T - CPO*TO - CLS*TLS$$

where

TS : total sales in period T (units)

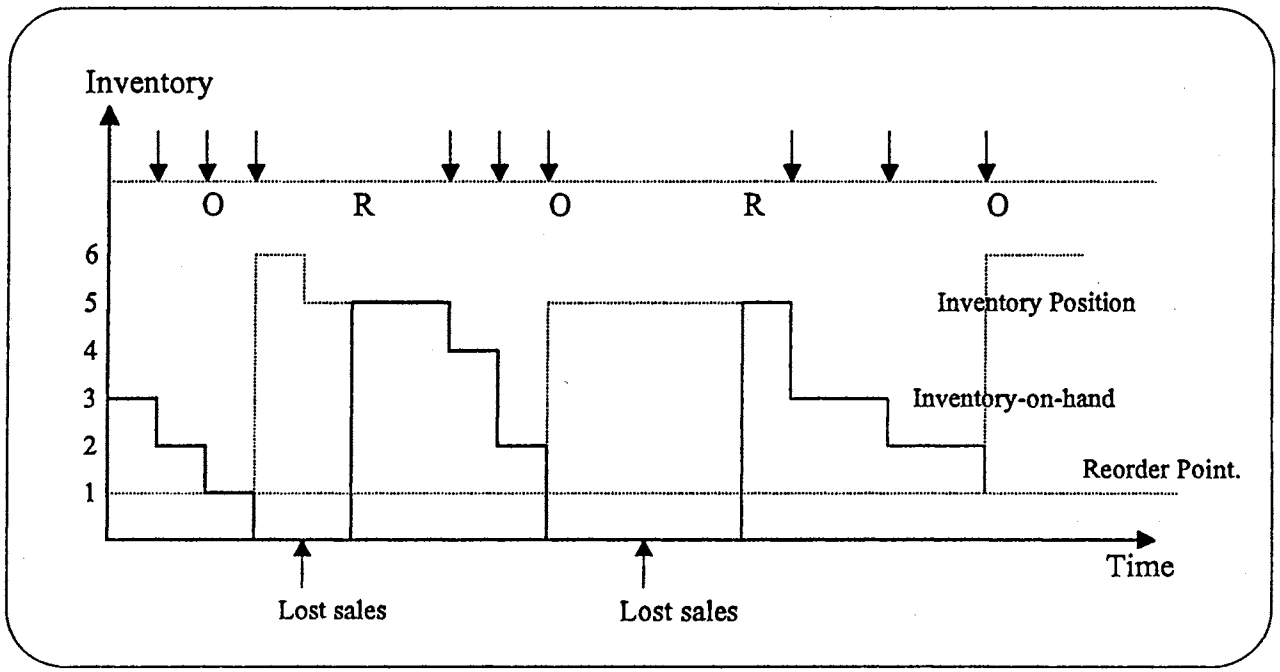
PPU : price per unit ($/unit)

CPU : cost per unit ($/unit)

TO : total ordered inventory in period T (units)

CPO : cost per order ($/order)

TO : total orders in period T



Legend:
↓ arrival of Demand
O - order 5 units
R - receive 5 units

Figure 26. Behavior of Continuous Review Inventory System

HC : holding cost of inventory ($/unit time)

ASV : average inventory on hand in period T (units)

CSL : cost per lost sale ($/unit)

TLS : total lost sales in period T (unit)

T : time interval

TS*PPU :total revenue

TO*CPU : total inventory cost

H*ASV*T : total holding cost

CLS*TLS : total cost of lost demand

CPO*TO : total cost of ordering

The following performance variables was used to calculate profit.

- level of inventory on hand

- number of sales

- number of orders

- number of lost sales

Although it is implicit, these performance variables were functions of the reorder point and reorder quantity. The initial inventory on hand was considered constant and could be set at any value between zero and infinity. Figure 27 shows the time plot of the cumulative average profit for the continuous review system in which no backorders were allowed.

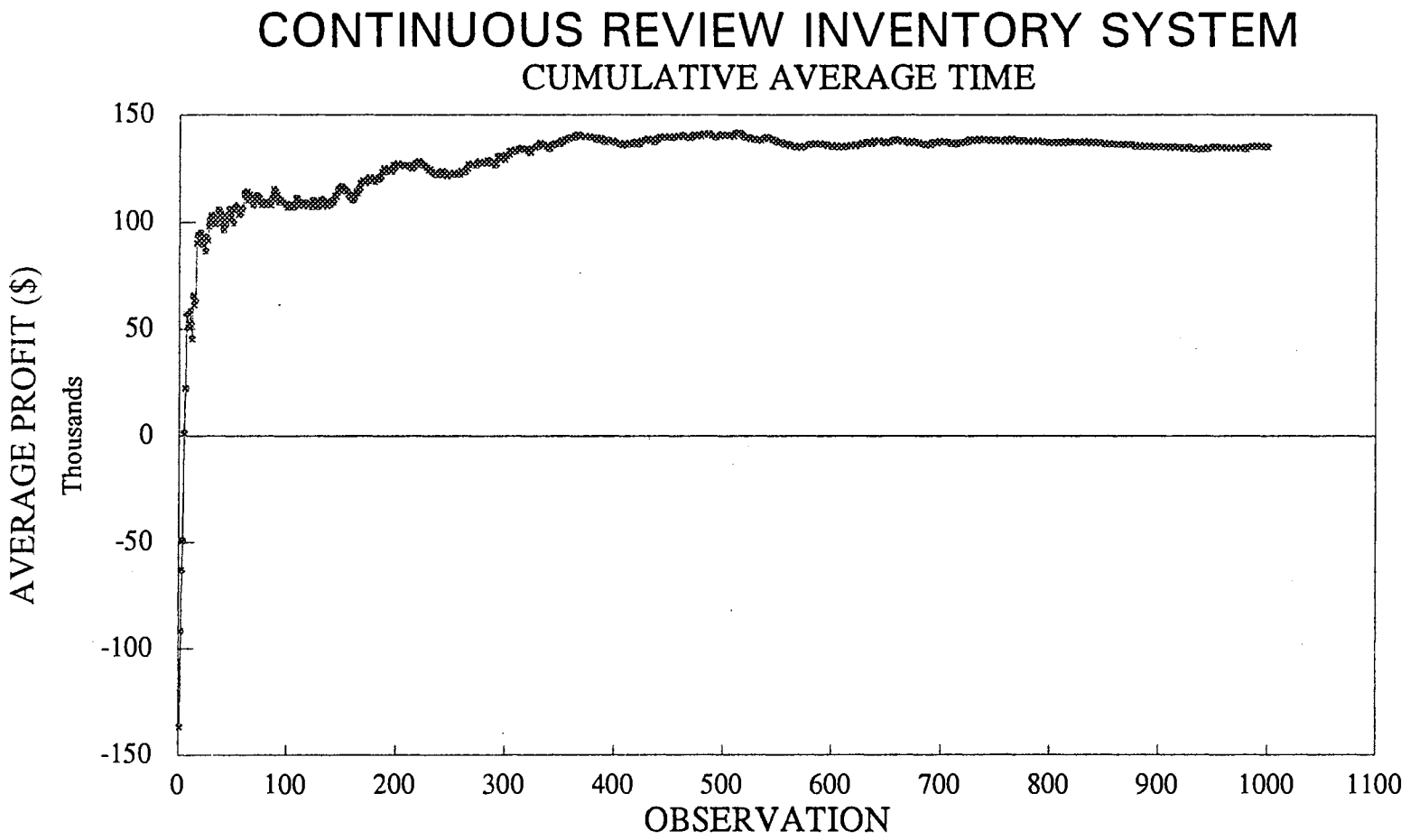
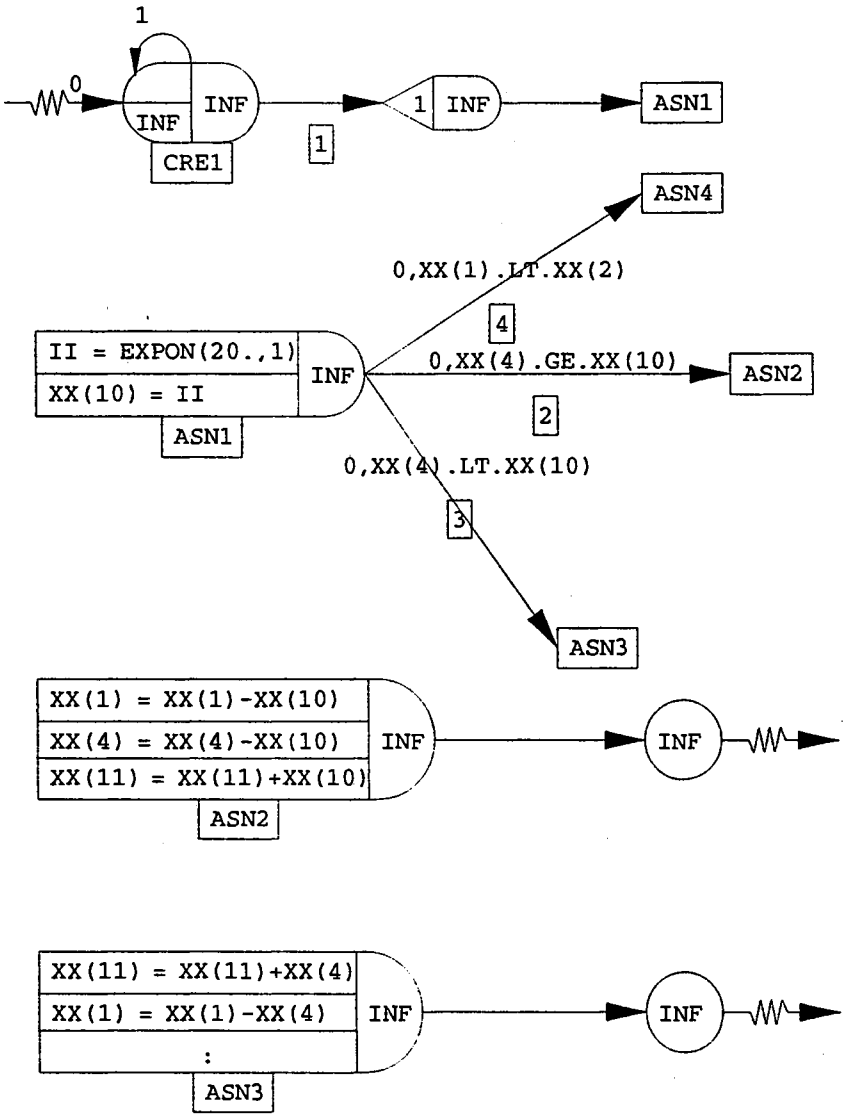
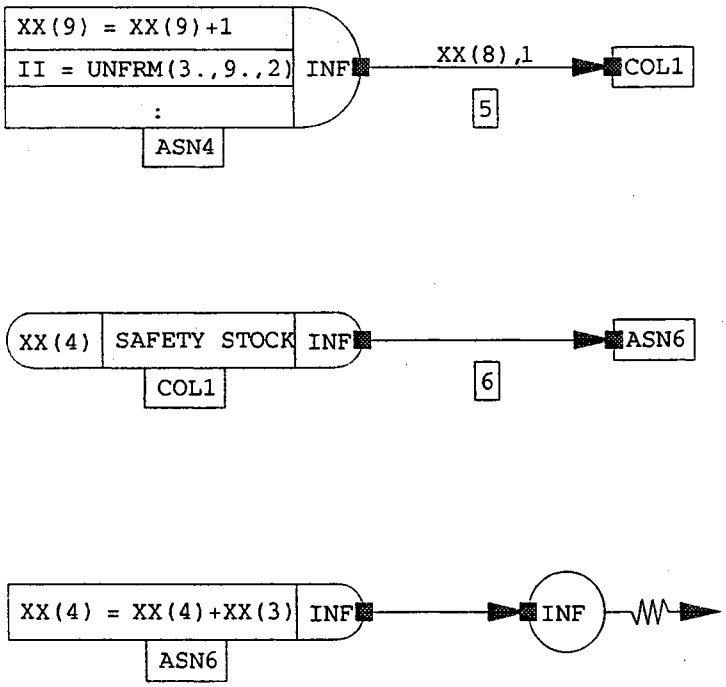


Figure 27. Cumulative Average Profit Plots for Inventory System

APPENDIX G

SLAM NETWORK FOR THE CONTINUOUS REVIEW INVENTORY PROBLEM


1
0
INF
INF
CRE1
1
1
INF
ASN1
ASN4
0,XX(1).LT.XX(2)
4
II = EXPON(20.,1)
XX(10) = II
ASN1
INF
0,XX(4).GE.XX(10)
ASN2
2
0,XX(4).LT.XX(10)
3
ASN3
XX(1) = XX(1)-XX(10)
XX(4) = XX(4)-XX(10)
XX(11) = XX(11)+XX(10)
ASN2
INF
INF
XX(11) = XX(11)+XX(4)
XX(1) = XX(1)-XX(4)
:
ASN3
INF
INF


XX(9) = XX(9)+1
II = UNFRM(3.,9.,2)
:
ASN4
INF
XX(8),1
COL1
5
XX(4)
SAFETY STOCK
COL1
INF
ASN6
6
XX(4) = XX(4)+XX(3)
ASN6
INF
INF

APPENDIX H

USER'S GUIDE

USER'S GUIDE

This appendix is designed to provide the user with useful information about the developed system. The automated simulation optimization system (ASOPT) is composed of a main program and 32 subroutines. Figure 28 shows the structure of the system. Brief descriptions of each module and submodule are given on the following pages. These descriptions provide information about the purpose of the module and other modules which call and are called by it. The developed system permits a maximum of 10 controllable factors to be considered in a search for a maximum or a minimum simulation response. Usage of the simulation optimization system should require only a minimal amount of effort on the part of the user. Two relatively easy tasks must be accomplished:

1. The system must be interfaced with the simulation to which it is to be applied.

2. The necessary input must be provided.

In order to achieve the desired results, a successful interface of the simulation program with the system is a must. The interface can be characterized by four basic areas:

- inter-program communication,
- simulation input considerations,
- factor range definitions,
- requirements for computer resources.

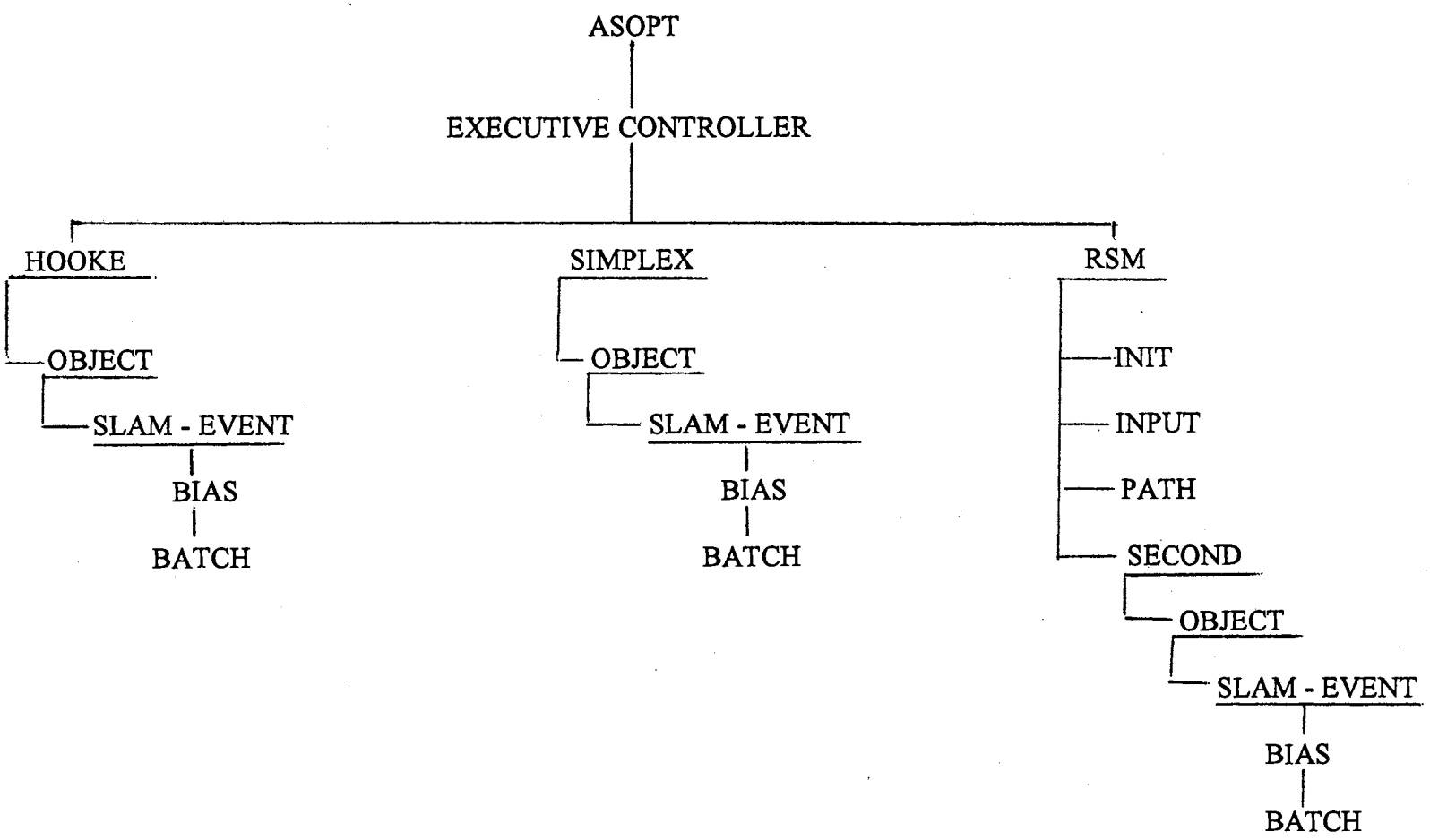


Figure 28. Structure of the Automated Simulation Optimization Program

Table IX. ASOPT Modules and Their Functions

PROGRAM	PURPOSE	CALLS	CALLED BY
BATCH	Determine the steady state	-----	EVENT-Slam
BIAS	Detect the initial bias in the output data	----	EVENT-Slam
BMATRX	Form the Bmatix and returns its largest eigen values	LOCATE EIGEN CENTER	SECOND
DONE	Print messages, results and ends the search	----	INPUT FIRST PATH
EIGEN	Find eigen values of a matrix	CENTER	BMATRX
FACTOR	Construct fractional factorial design	-----	FIRST
FIRST	Fit first-order surface	FACTOR RESTAR SIMUL WRITER DONE	RSM
HOOKE	Find the minimum of a multivariable uncon-strained nonlinear objective function	OBJECT	MAIN
INIT	Initailize the data storage	------	RSM
INPUT	Read the input data	SIMUL RESTAR DONE WRITER	RSM
OBJECT	Call the simulation	----	SIMUL

Table IX. ASOPT modules and their functions (continued)

PROGRAM	PURPOSE	CALLS	CALLED BY
ORDER	Put the elements of matrix in descending order	----	PATH
PATH	Perform experiments along the path of steepest ascent	SIMUL ORDER DONE	RSM
QUADR	Obtain the coefficients of a second order model	----	SECOND
RESTAR	Restart the program	----	INPUT FIRST PATH SECOND
RSM	Conduct the search	INIT INPUT FIRST PATH SECOND	MAIN
SECOND	Fit a second order design	SIMUL QUADR BMATRX	RSM
SIMPLEX	Find the minimum of a objective function using the Nelder and Mead Simplex algorithm	OBJECT	MAIN
SIMUL	Obtain the simulation response	SIM	INPUT FIRST PATH SECOND CSTOP
WRITER	Print the values of factors to the user chosen media	----	FIRST INPUT

Inter-Program Communication : In order to use the simulation optimization program in conjunction with the simulation program, the simulation program must be called from main program. Regardless of the type of the routine which might call the simulation, the simulation program must be called from the subroutine "OBJECT". It is the user's responsibility to make any necessary modifications , if a different simulation language is desired to be used instead of SLAM. Communication between the routines and the OBJECT routine is achieved by I/O operations.

Simulation Input Considerations: All inputs are passed via data files. Input to SLAM is a data file "SIN.DAT" which is written by an optimization routine and read by SLAM. The first 10 XX(I) SLAM variables are reserved for input purposes. Therefore, the user must arrange or change other input elements within the simulation so that the controllable factor values assigned by the simulation optimization system are not altered. The user must be certain that the simulation program does not destroy the values of the controllable factors.

The response obtained from each simulation run is written to the file "SOUT.DAT" which is read by the optimization routines. All the files used in the system are direct access, formatted files.

Factor Range Definition: All of the programs assume that each controllable factor is continuous and real valued. A minor problem might be encountered if a factor should have an integer value. Some of the routines have been modified to handle integer results if it is supplied with the necessary information. In that case, the user must made the necessary arrangements to round up or round down to the nearest integer and experiment with these input values.

Requirements for the Computer Resources: All of the programs were implemented and executed on a 486-33 Mhz IBM compatible personnel computer. The programs were written in Microsoft FORTRAN version 5.1 and simulation programs were written in SLAM and run on SLAMSYSTEM version 2.0.

Input-Output Requirements and Options: The user must input all data through the keyboard. Once all of the input are entered, the data is verified by printing all of the entries on the screen. After verification or modification by the user, the input data are written into the file "SIN.DAT" to be read by the simulation program.

Communication between the simulation program and the optimization modules is achieved by writing to and reading from the data file "SOUT.DAT". Finally, the results of the optimization modules are written into the file "OPT.DAT". Those three files are the minimum requirements of the successful execution of the automated simulation optimization system. The user must take necessary precautions when using those files which are direct access, formatted files. Other than those files, system messages were printed on the screen whenever they were necessary. Due to I/O restrictions of the FORTRAN language and long simulation response times, user interaction was minimized by limiting the number of "read from screen" statements.

Input to the system is done in two steps; - numeric input(e.g., number of variables, initial variables, etc.) and answers to the questions asked by the executive controller (needed only if ASOPT is used).

The output contains information about the optimum response, starting point, optimal points, number of function evaluations, total response time and average response

time. The user must modified the necessary routines, if he/she wants to customize standard system output, or needs extra information about the system.

APPENDIX I

COMPUTER LISTINGS

```
C*****************************************************************
C**                                                             **
C**                  AUTOMATED DISCRETE EVENT                   **
C**               SIMULATION OPTIMIZATION SYSTEM                **
C**                                                             **
C*****************************************************************
    COMMON /PRG/X(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
    CHARACTER CHR*9,MET(3)*30
    DATA MET/'RESPONSE SURFACE METHOD','NELDER & MEAD SIMPLEX',
   &'HOOKE & JEEVES'/
     OPEN(40,FILE='SIN.DAT',ACCESS='DIRECT',FORM='FORMATTED',
   &RECL=20)
    CHR='MINIMIZED'
    WRITE(6,10)
 10  FORMAT(T10,51('*'),/T10,'*',T60,'*'/T10,'*',T15,
   &40HAUTOMATED SIMULATION OPTIMIZATION SYSTEM,T60,'*'/T10,
   &'*',T60,'*'/T10,51('*'))
    PRINT*,'** ENTER 1 FOR OPTIMIZATION'
    PRINT*,'**          2 FOR STATISTICAL ANALYSIS'
    READ*,IANS
    IF(IANS.EQ.2) GO TO 500
    PRINT*,'** ENTER NUMBER OF VARIABLES'
    READ*, NVAR
    PRINT*,'** ENTER TYPE OF THE OPTIMIZATION'
    PRINT*,'  1- MINIMIZATION'
    PRINT*,'  0- MAXIMIZATION'
    READ*,MIN
    PRINT*,'** ENTER NUMBER OF SIMULATION RUNS ALLOWED'
    PRINT*,'  0 - CPU TIME GIVEN'
    READ*,N
    IF(N.EQ.0) THEN
       PRINT*,'ENTER CPU TIME'
       READ*,CPU
    ENDIF
    WRITE(6,11)
 11  FORMAT(///,T10,51('*')/,T10,'*',5X,'AVAILABLE METHODS',T60,
   &'*',/T10,51('*'))
    WRITE(6,12)
 12  FORMAT(T10,'*', 2X,'1 - RESPONSE SURFACE METHODOLOGY',T60,'*',
   &/T10,'*',2X,'2 - NELDER AND MEAD SIMPLEX METHOD',T60,'*',
   &/T10,'*',2X,'3 - HOOKE AND JEEVES PATTERN SEARCH',T60,'*',
   &/T10,51('*'))
    PRINT*,'** ENTER 1- AUTOMATIC OPTIMIZATION'
    PRINT*,'       2- USER SELECTED METHOD'
    READ*,MUSER
    IF(MUSER.EQ.2) THEN
```

```
      PRINT*,'** ENTER SELECTED METHOD NUMBER'
      READ*,METNUM
    ENDIF
99  DO 15 I=1,NVAR
      WRITE(6,16) '** ENTER INPUT VARIABLE(',I,') '
16     FORMAT(T2,A,I1,A,\)
      READ*, X(I)
15  CONTINUE
100 WRITE(*,17)
17  FORMAT(///,T10,51('+'),/T10,'+',10X,'INPUT SUMMARY',T60,
   &'+',/T10,51('+'))
    IF(MIN.EQ.0) CHR='MAXIMIZED'
    WRITE(*,18) CHR
18  FORMAT(T2,'1',T20,'OBJECTIVE FUNCTION WILL BE ',A)
    WRITE(6,19)NVAR
19  FORMAT(T2,'2',T20,'NUMBER OF VAIABLES IS ',I2)
    DO 20 I=1,NVAR
      WRITE(6,21) 2+I,'INPUT VARIABLE(',I,') = ',X(I)
21     FORMAT(T2,I1,T20,A,I1,A,F8.4)
20  CONTINUE
    IF(MUSER.EQ.2) THEN
      WRITE(6,22)3+NVAR,'USER SELECTED ',MET(METNUM)
    ELSE
      WRITE(6,22)3+NVAR,'EC SELECTED ',MET(METNUM)
22     FORMAT(T2,I1,T20,A,A )
    ENDIF
    WRITE(6,23)
23  FORMAT(//T10,'** PLEASE CHECK THE INPUT **',/T10,'** ENTER
   & NUMBER OF CORRECTIONS AND LINE NUMBERS')
    READ*,NERROR,LNUM
    IF(NERROR.GT.0) THEN
      IF(LNUM.EQ.1) THEN
        PRINT*,'ENTER OBJ. FUNCTION TYPE'
        READ*,MIN
      ELSEIF(LNUM.EQ.2) THEN
        PRINT*,'ENTER NUMBER OF VARIABLES'
        READ*,NVAR
        GO TO 99
      ELSEIF(LNUM.EQ.(3+NVAR)) THEN
        PRINT*,'ENTER THE METHOD CHOOSEN'
        READ*,METNUM
      ELSE
        PRINT*,'ENTER VARIABLE',LNUM-2
        READ*,X(LNUM-2)
      ENDIF
      GO TO 100
```

```
      ENDIF
      IF(MUSER.EQ.1) THEN
        CALL GETTIM(IHOUR,IMIN,ISEC,IHSEC)
        PS=IHOUR*3600+IMIN*60+ISEC+REAL(IHSEC)/100.
        CALL RUNSIM
        CALL GETTIM(IHOUR,IMIN,ISEC,IHSEC)
        PE=IHOUR*3600+IMIN*60+ISEC+REAL(IHSEC)/100.
        DIF=PE-PS
        NP=CPU/DIF
        PRINT*,'** PILOT RUN TAKES',DIF,'SECONDS'
        PRINT*,'** WITH THE GIVEN TIME YOU CAN MAKE AVERAGE',NP,' R
     %UNS'
        PRINT*,'** DO YOU WANT TO CHANGE TIME LIMIT'
        PRINT*,'** 1-YES  0-NO'
        READ*,ITIME
        IF (ITIME.EQ.1) THEN
          PRINT*,'MAXIMUM ALLOWED COMPUTER TIME'
          READ*,CPU
        ENDIF
        CALL RULE(NVAR,N,CPU,METNUM)
      ENDIF
      GO TO (111,222,333)METNUM
  111 CALL RSM
      IF(IRID.EQ.1) THEN
        IF(NREM.GT.2*NVAR.OR.CPU.GT.2*NVAR*DIF) THEN
          PRINT*,'** SOLUTION IS OUT OF EXPERIEMTAL REGION'
           PRINT*,**  NEED TO SWITCH ANOTHER CONSTRAINED METHOD'
*           CALL HOOKE
        ELSE
          PRINT*,'** NOT ENOUGH COMPUTER TIME LEFT'
        ENDIF
      ENDIF
      GO TO 666
  222 CALL SIMPLEX
      GO TO 666
  333 CALL HOOKE
  666 GOTO(110,220,330)ITERM
  110 PRINT*,'**  SUCESSFUL TERMINATION'
      IF(CPU.GT.2*NVAR*DIF) THEN
        PRINT*,'**WOULD YOU LIKE TO EXPLORE SURFACE'
        READ*,IANS
        IF(IANS.EQ.1) THEN
          DO 115 I=1,NVAR
  115         READ(54,*)X(I)
          CALL RSM
        ENDIF
```

```
      ENDIF
      GO TO 999
  220 PRINT*,'** FORCED TERMINATION DUE TO UNSUFFICIENT
     * RESOURCES'
      PRINT*,'** WOULD YOU LIKE TO CONTINUE? 1-YES 0-NO'
      READ*,IANS
      IF (IANS.EQ.1) THEN
        PRINT*,'WOULD YOU LIKE TO RESTART THE ALGORITHM 1-YES 0-NO'
        READ*,IANS
        IF(IANS.EQ.1) THEN
          PRINT*,'** RESTART FROM THE BEST POINT FOUND SOFAR 1-YES'
          READ*,IANS
          IF(IANS.EQ.1) THEN
            UBEST=1
          ELSE
            PRINT*,'**** ENTER INPUT VARIABLES'
            DO 138 I=1,NVAR
              READ*,X(I)
  138         CONTINUE
          ENDIF
          GO TO (88,89,90)METNUM
   88       CALL RSM
           GO TO 999
   89       CALL SIMPLEX
           GO TO 999
   90       CALL HOOKE
           GO TO 999
        ELSE
          UBEST=1
          GOTO(95,96,97) METNUM
   95      CALL SIMPLEX
          GO TO 999
   96      CALL HOOKE
          GO TO 999
   97      IF(NVAR.EQ.1) THEN
            CALL HOOKE
          ELSE
            CALL SIMPLEX
          ENDIF
        ENDIF
      ENDIF
      GO TO 999
  330 PRINT*,'**THE ALGORITH CONVERGES UNACCEPTABLY SLOW'
      PRINT*,'** WOULD YOU LIKE TO CONTINUE? 1-YES 0-NO'
      READ*,IANS
      IF(IANS.EQ.1.AND. CPU.GT.2*N*DIF) THEN
```

```
      PRINT*,'**** ENTER INPUT VARIABLES'
      DO 140 I=1,NVAR
        READ*,X(I)
140     CONTINUE
      GOTO(151,152,153)METNUM
151     CALL SIMPLEX
      GO TO 999
152     CALL HOOKE
      GO TO 999
153     IF(NVAR.GT.1) CALL SIMPLEX
    ENDIF
    GO TO 999
500 PRINT*,'** ENTER NUMBER OF VARIABLES'
    READ*,NVAR
    PRINT*,'** ENTER VARIABLES'
    READ*,(X(I),I=1,NVAR)
    WRITE(40,100,REC=1) REAL(NVAR)
     DO 550 I=1,NVAR
        WRITE(40,100,REC=I+1) X(I)
550  CONTINUE
    CALL RUNSIM
    PRINT*,'DO YOU WANT TO CONTINUE?   1-YES'
    READ*,IANS
    IF(IANS.EQ.1) GO TO 500
999  STOP
   END

   SUBROUTINE RULE(NVAR,N,CPU,METNUM)
   MAXTIM=604800
   PRINT*,'** DO YOU HAVE INFORMATION ABOUT THE RESPONSE
  & SURFACE? YES-1, NO-0'
   READ*,INFO
   IF(INFO.EQ.1) GOTO 500
100  IF(NVAR.EQ.1) THEN
      METNUM=3
      RETURN
   ENDIF
   IF(NVAR.LT.6) THEN
      IF(CPU.GE.MAXTIM) THEN
        METNUM=1
        RETURN
      ELSE
        METNUM=2
        RETURN
      ENDIF
   ELSE
```

```
      IF(CPU.GE.MAXTIM) THEN
        METNUM=1
        RETURN
      ELSE
        METNUM=3
      ENDIF
    ENDIF
    RETURN
500 PRINT*,'**POSSIBLE RIDGES ON THE RESPONSE SURFACE'
    PRINT*,'    1-YES    0-NO'
    READ*,IRID
    IF(IRID.EQ.1) THEN
      METNUM=3
      RETURN
    ENDIF
    PRINT*,'** IRREGULAR RESPONSE SURFACE 1-YES, 0-NO'
    READ*,ISUR
    IF(ISUR.EQ.1) THEN
      METNUM=2
      RETURN
    ENDIF
    PRINT*,'** FLAT RESPONSE SURFACE?  1-YES,  0-NO'
    READ*,IFLAT
    IF(IFLAT.EQ.1) THEN
      IF(NVAR.GT.1) THEN
        METNUM=2
      ELSE
        METNUM=3
      ENDIF
      RETURN
    ENDIF
    GO TO 100
    RETURN
    END

    SUBROUTINE RUNSIM
     OPEN(41,FILE='SOUT.DAT',ACCESS='DIRECT',FORM='FORMATTED',
    &RECL=20)
     I=SPAWNLP(0,LOC('INPUT'C),LOC('INPUT'C),INT(0))
     J=SPAWNLP(0,LOC('INVLOST.EXE'C),LOC('INVLOST.EXE'C),INT(0))
     K=SPAWNLP(0,LOC('OUTPUT'C),LOC('OUTPUT'C),INT(0))
     READ(41,50,rec=1)SUM
     PRINT *, 'SUM=',SUM
     READ(41,50,REC=4)PRERR
     IF(PRERR.EQ.1) THEN
        PRINT*,'!!! SYSTEM DID NO REACH STEADY STATE!!!'
```

```
      PRINT*,' DO YOU WANT TO INCREASE RUN LENGTH'
      READ*,IANS
      IF(IANS.EQ.1) THEN
              CALL CHANGE
     ELSE
              STOP
      ENDIF
    ENDIF
    RETURN
    END

    SUBROUTINE CHANGE
$INCLUDE: 'INTSLM.FI'
    INTEGER*2 SPAWNLP
    CHARACTER*30 A(20),TEMP,FNAME
    PRINT*,'ENTER NAME OF THE CONTROL FILE'
     READ(*,'(A)')FNAME
    OPEN(52,FILE=FNAME,STATUS='UNKNOWN')
    PRINT*,'ENTER THE CONTROL STATEMENT TO BE CHANGED'
    PRINT*,'******   CHECK THE CONTROL STATEMENT   ******'
    PRINT*,'****** SLAM CAN NOT DETECT ANY ERRORS !! ******'
    READ(*,'(A)')TEMP
    L=LEN_TRIM(TEMP)
    K=INDEX(TEMP(:L),',')
    DO 5 J=1,K-1
      IF (ICHAR(TEMP(J:J)).GT.96) THEN
        II=ICHAR(TEMP(J:J))-32
        TEMP(J:J)=CHAR(II)
      ENDIF
 5   CONTINUE
    I=1
 10   READ(2,'(A)',end=100) A(I)
      PRINT *,A(i)
      N=INDEX(A(I),',')
      IF(N.EQ.0) GO TO 9
      IF(N-1.LT.K-1) THEN
        M=N-1
      ELSEIF(N-1.GT.K-1) THEN
        M=K-1
      ELSE
        M=N-1
      ENDIF
C
C *** CHECK THE STRING
C
      IF (A(I)(1:M).EQ.TEMP(1:M))THEN
```

```
C
C *** REPLACE THE CONTROL STATEMENT
C
        A(I) = TEMP
      ENDIF
 9   I= I+1
     GO TO 10
 100  PRINT*,'END OF FILE REACHED'
     REWIND 2
     DO 20 J=1,I-1
        WRITE(52,'(A)') A(J)
 20   CONTINUE
     REWIND 52
     CLOSE (52,STATUS='KEEP')
     I=SPAWNLP(0,LOC('INPUT'C),LOC('INPUT'C),INT(0))
     J=SPAWNLP(0,LOC('BASECASE'C),
    & LOC('BASECASE'C),INT(0))
     K=SPAWNLP(0,LOC('OUTPUT'C),LOC('OUTPUT'C),INT(0))
     END
```

```
C*********************************************************************
C**  THIS SUBROUTINE PERFORMS HOOKE AND JEEVE'S PATTERN          **
C**   SEARCH (MODIFIED FROM KUESTER AND MIZE (1973))             **
C*********************************************************************
      SUBROUTINE HOOKE
      INTERFACE TO FUNCTION SPAWNLP[C,VARYING] (MODE)
      INTEGER*2 MODE
      END
      COMMON /PRG/RK(10),UBEST,IRID,NST,CPU,MIN,ITERM,NREM
      INTEGER*2 SPAWNLP
      DIMENSION EPS(10), Q(10), QQ(10), W(10),BETA(2)
      OPEN(40,FILE='SIN.DAT',ACCESS='DIRECT',FORM='FORMATTED',
     &RECL=20)
      OPEN(41,FILE='SOUT.DAT',ACCESS='DIRECT',FORM='FORMATTED',
     &RECL=20)
      OPEN(43,FILE='OPT.DAT',STATUS='UNKNOWN')
      PRINT *, 'ENTER 1 FOR INTERMEDIATE RESULTS, OTHERWISE ENTER 0'
      READ *,IP
      PRINT *, 'ENTER NUM. OF MAX. ITER. AND NUM. OF REDUCTION IN
     * STEP SIZE'
      READ *, MAXIT, NCUT
      PRINT*, 'ENTER THE VECTOR OF INITIAL STEP SIZE FOR EACH INPUT'
      PRINT*,' RECOMMENDED 1/10 OF A PARAMETER'
      READ *, (EPS(J), J=1,NST)
      PRINT *,'ENTER ALPHA, AND EPSILON (RECOMMENDED ALPHA=1'
      READ *, ALPHA, EPSY
      PRINT*,'ENTER BETA FOR EVERY VARIABLE (1 FOR INTEGERS)'
      READ*,(BETA(I),I=1,NST)
      QD=0.0
  100 FORMAT(F20.5)
      CALL GETTIM(IHOUR,MIN,ISEC,IHSEC)
      ST=3600*IHOUR+60*MIN+ISEC+REAL(IHSEC)/100.
      CALLHOOKE(RK,EPS,NST,MAXIT,NCUT,EPSY,ALPHA,BETA,QD,Q,QQ,
     * W,IP,IPTR,UBEST)
      CALL GETTIM(IHOUR,MIN,ISEC,IHSEC)
      PEND=3600*IHOUR+60*MIN+ISEC+REAL(IHSEC)/100.
      DIF=PEND-ST
      WRITE(43,*)'TOTAL RESPONSE TIME=',DIF
      AVGPT=DIF/REAL(IPTR-1)
      WRITE(43,*)'AVG. PROGRAM TIME=',AVGPT
      CPU=CPU-DIF
      NREM=N-IPTR
      END
```

```
      SUBROUTINE HOOKE(RK,EPS,NST,MAXIT,NCUT,EPSY,ALPHA,BETA,
     * QD,Q,QQ,W ,IP,IPTR,UBEST)
      DIMENSION RK(NST),EPS(NST),Q(NST),QQ(NST),W(NST),BETA(NST),
     & s(100,3)
      DATA S,SSAVE/300*0,0/
      KFLAG= 0
2     WRITE(40,100,REC=1)REAL(NST)
      DO 10 I=1,NST
      Q(I)=RK(I)
      W(I)=0.0
      WRITE(40,100, REC=I+1)abs(RK(I))
10    CONTINUE
100   FORMAT(F20.5)
      KAT=0.0
      KK1=0
      IPTR=1
      ITER=0
70    KCOUNT=0
      ITER=ITER+1
      WBEST= W(NST)
      CALL SEARCH(RK,IPTR,SUM,SFLG,S)
      IF(SFLG.EQ.1) THEN
         SSAVE=SSAVE+1
         GO TO 71
      ENDIF
      CALL OBJECT(SUM)
      S(IPTR,1)=RK(1)
      S(IPTR,2)=RK(2)
      S(IPTR,3)=SUM
      IPTR=IPTR+1
71    WRITE(43,*)SUM,RK(1),rk(2),kk1,SFLG
      KK1 = KK1 + 1
      BO= SUM
      IF(MOD(ITER,4).EQ.1)PRE=SUM
      IF(MOD(ITER,4).EQ.0) THEN
        PROG=(PRE-BO)/ABS(PRE)
        IF(PROG.LT.0.5) THEN
          ITERM=3
          GO TO 97
        ENDIF
      ENDIF
      IF( KK1.EQ.1) QD =SUM
      IF(KK1.EQ.1)  GO TO 201
      IF (BO.GT.QD)THEN
         KFLAG = 1
      ELSE
```

```
      QD= BO
    END IF

C**** SEARCH STARTS *****
C
 201  DO 55 I=1, NST
      QQ(I) = RK(I)
      TSRK = RK(I)
      RK(I) = RK(I) + EPS(I)
      WRITE(40,100,REC=I+1)abs(RK(I))
      WRITE(43,*) RK(I)
      PRINT*,RK(1),RK(2)
      CALL SEARCH(RK,IPTR,SUM,SFLG,S)
      IF(SFLG.EQ.1) THEN
        SSAVE=SSAVE+1
        GO TO 54
      ENDIF
      CALL OBJECT( SUM)
      S(IPTR,1)=RK(1)
      S(IPTR,2)=RK(2)
      S(IPTR,3)=SUM
      IPTR=IPTR+1
 54   WRITE(43,*) SUM,RK(1),RK(2),KK1,SFLG
      KK1 = KK1 + 1
      W(I) = SUM
      IF ( W(I).LT. QD) GO TO 58
      RK(I) = RK(I) - 2.0 * EPS(I)
      WRITE(40,100,REC=I+1)abs(rk(i))
      WRITE(43,*) RK(I)
      PRINT*,RK(1),RK(2)
      CALL SEARCH(RK,IPTR,SUM,SFLG,S)
      IF(SFLG.EQ.1) THEN
        SSAVE=SSAVE+1
        GO TO 53
      ENDIF
      CALL OBJECT (SUM)
      S(IPTR,1)=RK(1)
      S(IPTR,2)=RK(2)
      S(IPTR,3)=SUM
      IPTR=IPTR+1
 53    WRITE(43,*) SUM,RK(1),RK(2),KK1,SFLG
      KK1 = KK1 + 1
      W(I) = SUM
      IF ( W(I).LT. QD) GO TO 58
      RK(I) = TSRK
      WRITE(40,100,REC=I+1)abs(rk(i))
```

```
      WRITE(43,*) RK(I)
      KCOUNT = KCOUNT + 1
      IF (I.EQ.1) THEN
         W(I) = BO
      ELSE
         W(I) = W(I-1)
      END IF
      GO TO 55
58     QD = W(I)
      QQ(I) = RK(I)
55  CONTINUE
    WRITE(43,*)(RK(I),I=1,NST)
    IF (IP) 60,65,60
60  PRINT *, KK1
    PRINT *, ( RK(I), I=1, NST)
    PRINT*,'** DO YOU WANT TO CONTINUE? 0-NO'
    READ*,IANS
    IF(IANS.EQ.0) THEN
       STOP
    ELSE
       PRINT*,'**DO YOU WANT TO CHANGE PARAMETERS? 1-YES'
       READ*,IANS
       IF(IANS.EQ.1) THEN
              PRINT*,'START ALL OVER? 1-YES'
             READ*,IA
             IF(IA.EQ.1) THEN
                    PRINT*,'ENTER PARAMETERS'
                   READ*,(RK(I),I=1,NST)
                  GO TO 2
              ELSE
                    PRINT*,'ENTER 1 STEP SIZE, 2 ALPHA, 3 BETA'
                   GO TO (301,302,303)
301                 PRINT*,'ENTER STEP SIZE'
                    READ*,(EPS(I),I=1,NST)
                    GO TO 65
302                 PRINT*,'ALPHA'
                   READ*,ALPHA
                    GO TO 65
303                 PRINT*,'BETA FOR EACH PARAMETER'
                    READ*,(BETA(I),I=1,NST)
              ENDIF
       ENDIF
    ENDIF
C
C   TEST FOR THE TERMINATION
C
```

```
 65  IF (KK1.GT.MAXIT) GO TO 94
     IF ( KAT. GE. NCUT) GO TO 94
     IF ( ABS(W(NST)-WBEST) .LE. EPSY) GO TO   95
C
C **** IF ALL AXIS FAIL, REDUCE THE STEP SIZE ***
C
     IF (KCOUNT.GE .NST) GO TO 28
     DO 26 I= 1,NST
        RK(I) = RK(I) + ALPHA * (RK(I)- Q(I))
        IF(RK(2).LE.0) RK(2)=1
        WRITE(40,100,REC=I+1)abs(RK(I))
 26  CONTINUE
     WRITE(43,*)(RK(I),I=1,NST)
     DO 25 I = 1, NST
 25      Q(I) = QQ(I)
     GO TO  70
 28  KAT = KAT +1
     IF (KFLAG. EQ. 1) GO TO 202
     GO TO 204
 202 KFLAG= 0
     DO 203 I=1,NST
        RK(I) = Q(I)
        WRITE(40,100,REC=I+1)abs(RK(I))
 203 CONTINUE
     WRITE(43,*)(RK(I),I=1,NST)
 204 DO 80 I=1,NST
     EPS(I) = EPS(I) * BETA(i)
 80  CONTINUE
     GO TO 70

C
C**** PRINT RESULTS   **
C
 94  ITERM=2
     GO TO 97
 95  ITERM=1
     GO TO 97
 97  WRITE (43,11) (EPS(I), i=1, NST)
 11  FORMAT(1X,20HTHE FINAL EPS ARE : , 4F20.8/)
     WRITE(43, 12) (RK(I),I=1,NST)
 12  FORMAT (1X, 20HTHE FINAL RK ARE    , 4F20.8/)
     WRITE(43,13) QD
 13  FORMAT ( 1X, 25HTHE MINIMUM REPONSE IS  :,F20.8/)
     WRITE(43,14)KK1
 14  FORMAT(1X, 32HNUMBER OF FUNCTION EVALUATIONS =,I4)
     DO 15 I=1, NST
```

```
      PRINT *, I,RK(I)
15  CONTINUE
    WRITE(43,16) INT(SSAVE)
16  FORMAT(1X,'NUMBER OF SEARCHES SAVED =',I3)
    RETURN
    END


C
C***  OBJECTIVE FUNCTION
C
    SUBROUTINE OBJECT (SUM)
C
C **  CALL SLAM PROGRAM
C
    I=SPAWNLP(0,LOC('INPUT'C),LOC('INPUT'C),INT(0))
    J=SPAWNLP(0,LOC('INVLOST.EXE'C),LOC('INVLOST.EXE'C),INT(0))
    K=SPAWNLP(0,LOC('OUTPUT'C),LOC('OUTPUT'C),INT(0))
C
C **  WRITE RESULT OF SIMULATION TO A OUTPUT FILE
C
    READ(41,50,REC=4)ICHK
    IF(ICHK.EQ.1) GO TO 60
    READ(41,50,rec=1)SUM
    PRINT *, 'SUM=',SUM
    IF(MIN.EQ.0) SUM=-SUM
50  FORMAT(F20.5)
    RETURN
60  PRINT*,'**SYSTEM DID NOT REACH STEADY STATE'
     PRINT*,'NEEDS LONGER RUNS'
      STOP
     RETURN
    END
```

```
C*********************************************************************
C*      THIS SUBROUTINE SEARCHES PREVIOUS DATA POINTS TO             *
C* PREVENT REDUNDANT OBJECTIVE FUNCTION EVALUATIONS                  *
C*********************************************************************
   SUBROUTINE SEARCH(RK,IPTR,SUM,SFLG,S)
   REAL S(100,3) ,RK(2)
   SFLG=0
   DO 10 I=1,IPTR-1
     IF(RK(1).EQ.S(I,1).AND.RK(2).EQ.S(I,2)) THEN
       SFLG=1
       SUM=S(I,3)
       RETURN
     ENDIF
10  CONTINUE
   RETURN
   END
```

```
C*********************************************************************
C*  THIS SUBROUTINE PERFORMS NELDER AND MEAD SIMPLEX SEARCH          *
C*   (MODIFIED FROM KUESTER & MIZE (1973))                           *
C*********************************************************************
      SUBROUTINE SIMPLEX
     INTERFACE TO FUNCTION SPAWNLP[C,VARYING] (MODE)
     INTEGER*2 MODE
     END
     INTEGER*2 SPAWNLP
     COMMON /PRG/XX(10),UBEST,IRID,N,CPU,MIN,ITERM,NREM
     COMMON/N1/NUMBER,NSAVE,TOTRES
     DIMENSION X(3,2), XCEN(3,2), XREF(3,2), XCON(3,2),XEX(3,2),Z(3)
     INTEGER AP
     OPEN(40,FILE='SIN.DAT',ACCESS='DIRECT',FORM='FORMATTED',
    &RECL=20)
     OPEN(41,FILE='SOUT.DAT',ACCESS='DIRECT',FORM='FORMATTED',
    &RECL=20)
     OPEN(43,FILE='OPT.DAT',STATUS='UNKNOWN')
     NUMBER=0
     NSAVE=0
     TOTRES=0
     WRITE(40,111,REC=1)REAL(N)
 111  FORMAT(F20.5)
     PRINT *,'ENTER MAXIMUM NUMBER OF ITERATIONS'
     READ*, MAXIT
     PRINT *, 'ENTER 1 FOR INTERMEDIATE 0 FOR FINAL RESULTS'
     READ *, IP
     PRINT*, 'ENTER REFLECTION, CONTRACTION AND EXPANSION
COEFFIENTS'
     PRINT*,'(--RECOMMENDED 1,.5,2)'
     READ *, ALFA, BETA, GAM
     PRINT*, 'ENTER CONVERGENCE PARAMETER'
     READ *, ACC
     PRINT *,'ENTER THE SIDE LENGTH OF SIMPLEX'
     READ *, A
     DO 112 J=1,N
        IF(UBEST.EQ.1) THEN
           X(1,J)=X(NP1,J)
        ELSE
           X(1,J)=XX(J)
        ENDIF
 112  CONTINUE
C
C *** SET INITIAL SIMPLEX ***
C
     Q = (A/(N* SQRT(2.0))) * ((N+1)**.5 -1)
```

```
      P =(A/(N*SQRT(2.0))) * ((N+1)**.5 + N-1)
      M = N+1
      DO 210 I = 2,M
         AP = 1
         DO 211  J=1,N
            AP = AP+1
            IF (AP .EQ. I) THEN
               X(I,J) = X(1,J) + P
            ELSE
               X(I,J) = X(1,J) + Q
            ENDIF
211      CONTINUE
210   CONTINUE
      NP1= N+1
      ITR = 0
      NFC=0
 250  DO 220 I = 1,NP1
         PRINT*,I,X(I,1),X(I,2)
         CALL OBJECT(I,X,Z,N,NP1)
         PRINT*,Z(I)
220   CONTINUE
      NFC=NFC+NP1
      ITR = ITR +1
      IF (ITR.GT.MAXIT) GO TO 295
      IF (IP)258,262,258
258   WRITE(*,221) ITR
221   FORMAT (/,2X,17HITERATION NUMBER ,I3)
      DO 222 J =1,NP1
222   WRITE(43,206)(J,I,X(J,I), I=1,N)
206   FORMAT(/,2(2X,2HX(,I2,1H,,I2,4H) = , E12.5))
      WRITE (43,207) (I,Z(I), I=1,NP1)
207   FORMAT(/,2X,2HF(,I2 ,4H) = ,E16.8)
      PRINT*,'DO YOU WANT TO CONTINUE  1-YES'
      READ*,IANS
      IF(IANS.EQ.1) GO TO 262
      ITERM=3
       STOP
       RETURN
C
C *** FIND MAXIMUM AND MINIMUM VALUES FOR OBJECTIVE FUNCTIONS
C
 262  ZHI = AMAX1(Z(1),Z(2),Z(3))
      ZLO = AMIN1(Z(1),Z(2),Z(3))
C
C *** FIND THE WORST POINT
C
```

```
      DO 265  I=1,NP1
        IF (ZHI.EQ.Z(I)) GO TO 270
  265 CONTINUE
  270 K=I
C
C *** CALCULATE CENTROID
C
      EN = N
      DO 271 J=1,N
        SUM = 0.0
        DO 272 I=1,NP1
          IF (K.EQ.I) GO TO 272
          SUM= SUM + X(I,J)
  272     CONTINUE
        XCEN(K,J) = SUM / EN
  271 CONTINUE
      I=K
      CALL OBJECT (I,XCEN,Z,N,NP1)
      NFC=NFC+1
      ZCEN = Z(I)
      SUM=0.0
      DO 273 I =1,NP1
        IF (K.EQ.I) GO TO 273
        SUM = SUM+(Z(I)-ZCEN)*(Z(I)-ZCEN)/EN
  273 CONTINUE
      EJ = SQRT(SUM)
      IF (EJ.LT.ACC)THEN
        ITERM=1
        GO TO 298
      ENDIF
      DO 274 J=1,N
        XREF(K,J) = XCEN(K,J) + ALFA*(XCEN(K,J) - X(K,J))
  274 CONTINUE
      I=K
      CALL OBJECT(I,XREF,Z,N,NP1)
      NFC=NFC+1
      ZREF = Z(I)
      DO 275  I=1,NP1
        IF (ZLO.EQ.Z(I)) GO TO 276
  275 CONTINUE
  276 L=I
      IF (ZREF.LE.Z(L)) GO TO 240
      DO 277 I=1,NP1
        IF (ZREF.LT.Z(I)) GO TO 278
  277 CONTINUE
      GO TO 280
```

```
278  DO 279 J=1,N
279      X(K,J) = XREF(K,J)
     GO TO 250
280  DO 281 J=1,N
281      XCON(K,J) = XCEN(K,J) + BETA*(X(K,J)-XCEN(K,J))
     I = K
     CALL OBJECT (I,XCON,Z,N,NP1)
     NFC=NFC+1
     ZCON=Z(I)
     IF (ZCON.LT.Z(K)) GO TO 285
     DO 284 J=1,N
        DO 284 I=1,NP1
           X(I,J) = (X(I,J)+X(L,J))/2.
284  CONTINUE
     GO TO 250
285  DO 287 J=1,N
        X(K,J) = XCON(K,J)
287  CONTINUE
     GO TO 250
240  DO 245 J=1,N
        XEX(K,J) = XCEN(K,J)+GAM*(XREF(K,J)-XCEN(K,J))
245  CONTINUE
     I = K
     CALL OBJECT(I,XEX,Z,N,NP1)
     NFC=NFC+1
     ZEX=Z(I)
     IF (ZEX.LT.Z(L)) GO TO 255
     DO 247 J=1,N
247      X(K,J) = XREF(K,J)
     GO TO 250
255  DO 260 J=1,N
260      X(K,J) = XEX(K,J)
     GO TO 250
295  WRITE(43,290) MAXIT
290  FORMAT(///,10X,'ALGORITHM DID NOT CONVERGE IN ',I5,'
ITERATIONS')
     ITERM=2
298  WRITE(43,291) ZLO
291  FORMAT(//,2X,'OPTIMUM VALUE OF F = ',E16.8)
     WRITE (43,292)
292  FORMAT(//,2X,'OPTIMUM VALUES OF VARIABLES ')
     DO 293 I=1,N
293      WRITE(43, 294) I,X(NP1,I)
294  FORMAT(/,2X,2HX9,I2,4H) = ,E16.8)
     PRINT*,'NUMBER OF EVALUATIONS=',ITR
     WRITE(43,*)'PROGRAM RESPONSE TIME',TOTRES
```

```
      AVGPR=TOTRES/REAL(NFC)
      WRITE(43,*)'AVERAGE PROGRAM RESPONSE TIME=',AVGPR
      WRITE(43,*)'NUMBER OF FUNC. EVALUATIONS=',NFC
      WRITE(43,*)'NUMBER OF CALLS TO OBJ. FUNC.=',NUMBER
      WRITE(43,*)'NUMBER OF SAVED CALLS TO OBJ. FUNC.=',NSAVE
      END

      SUBROUTINE OBJECT(II,X,Z,N,NP1)
      COMMON/N1/NUMBER,NSAVE,TOTRES
      DIMENSION X(NP1,N), Z(NP1),TEMP(300,3)
       FLG=0
       IX1=ABS(X(II,1))
       IX2=ABS(X(II,2))
      IF(NUMBER.LT.3) GO TO 20
      DO 10 J=1,NUMBER
        IF(IX1.EQ.TEMP(J,1).AND.IX2.EQ.TEMP(J,2))THEN
          Z(II)=TEMP(J,3)
          NSAVE=NSAVE+1
          RETURN
        ENDIF
   10  CONTINUE
   20   print*,'2','   number',number
       NUMBER =NUMBER+1
       TEMP(NUMBER,1)=ix1
       TEMP(NUMBER,2)=ix2
      WRITE(40,50,REC=1)REAL(N)
      WRITE(40,50,REC=2)REAL(IX1)
      WRITE(40,50,REC=3)REAL(IX2)
      PRINT*,'REALLY CALLING SLAM'
      CALL GETTIM(IHOUR,MIN,ISEC,IHSEC)
      PSTART=IHOUR*3600+MIN*60+ISEC+REAL(IHSEC)/100.
     I=SPAWNLP(0,LOC('INPUT'C),LOC('INPUT'C),INT(0))
      JJ=SPAWNLP(0,LOC('INVLOST.EXE'C),LOC('INVLOST.EXE'C),INT(0))
      K=SPAWNLP(0,LOC('OUTPUT'C),LOC('OUTPUT'C),INT(0))
C
C **  WRITE RESULT OF SIMULATION TO A OUTPUT FILE
C
      CALL GETTIM(IHOUR,MIN,ISEC,IHSEC)
      PEND=IHOUR*3600+MIN*60+ISEC+REAL(IHSEC)/100.
      DIF=PEND-PSTART
      TOTRES=TOTRES+DIF
      READ(41,50,REC=1)SUM
      PRINT *,'INSIDE FUNCTION','I=',II
      PRINT *, 'SUM=',SUM ,' ',X(II,1),X(II,2)
   50  FORMAT(F20.5)
      Z(II)=-SUM
```

```
TEMP(NUMBER,3)=Z(II)
RETURN
END
```

```
C*********************************************************************
C*   THIS SUBROUTINE USES RESPONSE SURFACE METHODOLOGY TO      *
C*   OPTIMIZE THE SIMULATION RESPONSE (SEE MYERS (1976))       *
C*   (MODIFIED FROM DENNIS E. SMITH (1976)                     *
C*********************************************************************
      SUBROUTINE RSM
      COMMON /PRG/XX(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
      COMMON /RSM1/X(15),Y
      COMMON /RSM2/LGSTR,LGSTK,IRSRT
      COMMON /RSM3/ALIST(220),L1(65),NDES,L2(6)
      COMMON /RSM4/IW1(4),IW2(4)
      COMMON /RSMC1/CLIST(450),LISTC(215)
      COMMON /RSMC2/CLST(4),LSTC(4)
C
C*** INITIALIZE DATA
C
      OPEN(51,FILE='RESULT.DAT',STATUS='UNKNOWN')
      CALL INIT
      CALL INPUT
      IF(IRSRT.NE.0) GOTO (100,101,102,103,103) IRSRT
  100 CALL FIRST
  101 CALL PATH
      IF(NDES.GE.2) GO TO 102
      CALL SUBSEC
      IF (NDES.LT.2) GO TO 100
  102 CALL SECOND
  103 CALL DONE(7)
      CALL RIDGE
      RETURN
      END


C*********************************************************************
C*     THIS SUBROUTINE INITAILIZES DATA STORAGE                  *
C*********************************************************************
      SUBROUTINE INIT
      COMMON /RSM1/X(15),Y
      COMMON /RSM2/LGSTR,LGSTK,IRSRT
      COMMON /RSM3/ALIST(220),LIST(72)
      COMMON /RSMC1/CLIST(450),LISTC(215)
      COMMON /RSMC2/CLST(4),LSTC(4)
      LGSTR=4
      LGSTK=15
      DO 10 I=1,15
   10    X(I)=0.0
      DO 11 I=1,220
```

```
11      ALIST(I)=0.0
   DO 12 I=1,72
12    LIST(I) =0.0
   DO 13 I=1,4
      CLST(I)=0
13     LSTC(I)=0
   DO 14 I=1,215
      LISTC(I)=0
      CLIST(I)=0
14  CONTINUE
   DO 15 I=216,450
      CLIST(I)=0
15  CONTINUE
   RETURN
   END
```

```
C*********************************************************************
C*   THIS SUBROUTINE READS THE INPUT DATA                            *
C*********************************************************************

      SUBROUTINE INPUT
      COMMON /PRG/XX(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
$INCLUDE:'RSMCOM.FI'
      PRINT*,'**  ENTER TOTAL NUMBER OF SIMULATION RUNS'
      READ *,N
      PRINT *,'ISTHIS A RESTART? 1-YES, 0-NO'
      READ *, IRSRT
      PRINT *,'ENTER NUMBER OF REPLICATIONS  (3 SUGGESTED)'
      READ*,M
      PRINT *,'ENTER TYPE OF THE OBJECTIVE FUNCTION 1-MIN, 0-MAX'
      READ *,MIN
      PRINT *,' ENTER NUMBER OF SIMULATION RUNS IN THIS PASS'
      READ *,NNOW
      PRINT *,'**  ENTER  1- TO WRITE TO THE SCREEN'
      PRINT *,'          2- TO WRITE TO THE FILE'
      PRINT *,'          3- BOTH'
      READ*,INPFLG
      K=NVAR
      PRINT*,'ENTER NUMBER OF CONSTRAINTS'
      READ*,NCST
      KT=K
      IF(IRSRT.EQ.0) GO TO 9
      NNOWS=NNOW
      CALL RESTAR(1)
      NNOW=NNOWS
      NVAL=0
 9    IF(IRSRT.NE.0) RETURN
      DO 10 I=1,LGSTK
 10     INDX(I) =I
      PRINT *,'ENTER EACH FACTOR AND ITS STEP SIZE'
      DO 11 I=1,KT
        PRINT *,'ENTER FACTOR',I,' STEP SIZE'
        READ *,DELT(I)
        X(I)=XX(I)
 11       CX(I) = X(I)
      CALL WRITER(INPFLG)
      FLM=M
C
C***  READ CONSTRAINTS
C
      DO 24 I=1,NCST
         PRINT*,'ENTER CONSTRAINT COEFFICIENTS'
```

```
      READ*,CZERO(I),(KVIDN(I,J),CCOEF(I,J),J=1,6)
24   CONTINUE
     DO 36 I=1,NCST
       J2=0
       DO 26 J=1,6
         IF(KVIDN(I,J).EQ.0) GO TO 28
         J2=J2+1
26       CONTINUE
28       WRITE(51,29)I,CZERO(I)
29       FORMAT(T10,'CONSTRAINT ',I2,5X,'A(0)=',F20.5)
       DO 34 J=1,J2
         WRITE(51,35)KVIDN(I,J),CCOEF(I,J)
35         FORMAT(10X,2HA(,I2,2H)=,F15.6)
34       CONTINUE
36   CONTINUE
C
C*** CHECK CONSTRAINT VIOLATIONS
C
     CALL STEP2(LITE,0)
     IF(LITE.EQ.0) GO TO 38
     CALL DONE(5)
38   DO 40 I=1,K
40     X(I)=CX(I)
     ISHFT=0
C
C**  OBTAIN OBJECTIVE FUNCTION VALUE FOR DESIGN CENTER
C
      CALL SIMUL
      YCENT=Y
     IF(NVAL.LT.NNOW) THEN
         RETURN
     ELSE
         IRSRT=1
         CALL RESTAR(2)
     ENDIF
     END
```

```
C*********************************************************************
C*    THIS SUBROUTINE PERFORMS THE FIRST PHASE OF THE                *
C*    RESPONSE SURFACE METHODOLOGY.  IT OBTAINS SIMULATION           *
C*    RESULTS CORRESPONDING TO POINTS IN THE FRACTIONAL              *
C*    FACTORIAL DESIGN                                               *
C*********************************************************************

      SUBROUTINE FIRST
$INCLUDE:'RSMCOM.FI'
      COMMON /RSMC2/ABFCT,GAMMIN,R,YSAVE,ISHFT,JGAM,NCST,NTIE
      IF(IRSRT.EQ.1) GO TO 14
C
C**  CONSTRUCT FRACTIONAL FACTORIAL DESIGN
C
      LFRST=0
      CALL FACTOR
      DO 10 I=1,LGSTK
 10     B(I)=0
      IF((N-NRUN).LT.(LN+2)) THEN
        CALL DONE(3)
      ENDIF
      IF(ISHFT.EQ.0) GO TO 8
      DO 6J=1,K
         I=INDX(J)
 6       X(I)=CX(I)
      B(IE)=0
      LFRST=0
      GO TO 12
C
C**  SET UP THE NEXT POINT IN THE DESIGN
C
  8   LFRST=1
 11   CALL FACTOR
 12   CALL SIMUL
      IF(NVAL.LT.NNOW) GO TO 14
      IRSRT=1
      CALL RESTAR(2)
 14   IF(LFRST.GT.0) GO TO 16
      YCENT=Y
      GO TO 20
C
C**  CALCULTE B(I)'S
C
 16   B(IE)=B(IE)+Y
      DO 18 I=1,MAXK
         SX=LX(I)
```

```
18      B(I)=B(I)+Y*SX
    IF(LFRST.GE.LN) GO TO 22
20  LFRST=LFRST+1
    IF(LFRST.NE.ISHFT) GO TO 11
    CALL FACTOR
    NRRN=NRRN+1
    Y=YSAVE
    GO TO 14
22  SSM=(FLM*(B(IE)+YCENT)**2)/(FLN+1.0)
    PRINT*,'*** FRACTIONAL FACTORIAL COMPLETED ***'
    write(51,55)
55  FORMAT(T10,'  *** FRACTIONAL FACTORIAL COMPLETED ***')
    RSQ=0.0
    DO 26 I=1,K
26      RSQ=RSQ+(FLM*B(I)**2)/FLN
    SS=(FLM*FLN*(YCENT-B(IE)/FLN)**2)/(FLN+1.)
    AMLF1=SS
    IDF=LN-K-1
    SS=0.0
    IF(IDF.NE.0) THEN
      IZ=K+1
      DO 30 I=IZ,MAXK
30        SS=SS+(FLM*B(I)**2)/FLN
      AMS=SS/REAL(IDF)
    ELSE
      AMS=0.0
    ENDIF
    AMLF2=AMS
    SIGER=0.0
    IF(M.EQ.1)GO TO 34
    SIGER=SIGSAV/((FLN+1.)*(FLM-1.))
34  RSQ=RSQ/(TSS-SSM)
    IF(SIGER.EQ.0.0) GO TO 38
    PRINT*,'*** ESTIMATE OF SIGMA OBTAINED FROM ITERATIONS***'
    GO TO 46
38  IF(SIGIN.GT.0.0) GO TO 42
    PRINT*,'*** DETERMINISTIC SIMULATION ASSUMED ***'
    WRITE(51,53)
53  FORMAT(T10,'*** DETERMINISTIC SIMULATION ASSUMED ***')
    GO TO 46
42  SIGER=SIGIN**2
    PRINT*,'*** INPUT ESTIMATE OF SIGMA USED ***'
46  PEMS = SIGER
    SIGER=sqrt(SIGER/FLM)
    SIGSAV=0.0
    PRINT*,'*** EST. STD. ERROR OF AVG. OBS. RESPONSE=', SIGER
```

```
      WRITE(51,58) SIGER
   58  FORMAT(T10,'*** EST. STD. ERROR OF AVG. OBS. RESPONSE=',F10.4)
      SIGSEC=SIGER
C
C*** STANDART ERROR OF  B(I)
C
      SIGB=(SIGER/SQRT(FLN))*2.0
      PRINT*,'&&&&& SIGMAB =',SIGB
C
C*** STEEPEST ASCENT PATH
C
      B(IE)=(B(IE)+YCENT)/(FLN+1.0)
      BHI=0.
      DO 50 I=1,K
        BLO=B(I)/FLN
        BHI=BHI+BLO**2
   50   B(I)=BLO
      SMBSQ=BHI
      IF(MAXK.EQ.K) GO TO 54
      IZ=K+1
      DO 52 I=IZ,MAXK
   52   B(I)=B(I)/FLN
   54  CALL WRITER(4)
      WRITE(51,56)
   56  FORMAT(T15,' B(I)s ARE  ')
      WRITE(51,57) B(IE),(B(I),I=1,K)
   57  FORMAT(T10,'B(0)=',F16.6,//(10X,F16.6))
      WRITE(*,*)B(IE),(B(I),I=1,K)
      END
```

```
C*************************************************************************
C*  THIS SUBROUTINE CONSTRUCTS FRACTIONAL FACTORIAL DESIGN   *
C*************************************************************************

      SUBROUTINE FACTOR
$INCLUDE:'RSMCOM.FI'
      DIMENSION JV(6)
      IF(LFRST.GT.1) GO TO 16
      IF(LFRST.EQ.1) GO TO 8
      J=1
      DO 10 I=1,LGSTR
         J=J+J
         IF(K.LT.J) GO TO 2
 10   CONTINUE
      STOP
 2    IR=I
      LN=J
      IF((K.EQ.3).AND.(N-NRUN).GT.6) GO TO 4
      GO TO 6
 4    IR=3
      LN=8
 6    FLN=LN
      MAXK=LN-1
      GO TO 32
 8    NRRN=0
      IE=0
      ISW=-1
      IRC=IR
      DO 12 I=1,LGSTR
         IB=IE+1
         IE=IE+IRC
         IF(MAXK.LT.IE) IE=MAXK
         DO 11 J=IB,IE
 11          LX(J)=ISW
         IF(IE.EQ.MAXK) GO TO 14
         ISW=-ISW
 12      IRC=IRC*(IR-I)/(I+1)
 14   IE=MAXK+1
      GO TO 28
 16   DO 18 I=1,IR
         LX(I)=-LX(I)
         IF(LX(I).EQ.1) GO TO 20
 18   CONTINUE
      STOP
 20   IF(MAXK.EQ.IR) GO TO 28
      NXT=IR+1
```

```
      I=2
      ISW=1
      J=0
      JC=0
   22 J=J+1
   24 JC=JC+1
      JV(J)=JC
      IF(LX(JC).LT.0) ISW=-ISW
      IF(I.GT.J) GO TO 22
      LX(NXT)=ISW
C
C***  SET UP THE NEXT ROW IN THE DEISGN
C
      IF(NXT.EQ.MAXK) GO TO 28
      NXT=NXT+1
   26 JC=JV(J)
      IF(LX(JC).LT.0) ISW=-ISW
      IF(JC.LT.(IR-I+J)) GO TO 24
      J=J-1
      IF(J.GT.0) GO TO 26
      I=I+1
      JC=0
      GO TO 22
C
C***  OBTAIN THE UNCODED VALUES
C
   28 DO 30 I=1,K
         J=INDX(I)
         X(J)=CX(J)-DELT(J)
         IF(LX(I).LT.0) GO TO 30
         X(J)=CX(J)+DELT(J)
   30 CONTINUE
   32 CONTINUE
      RETURN
      END
```

```
C*********************************************************************
C*   THIS SUBROUTINE DETERMINES THE PATH OF STEEPEST ASCENT      *
C*********************************************************************
      SUBROUTINE PATH
$INCLUDE:'RSMCOM.FI'
      IF(IRSRT.EQ.2) THEN
         GO TO 60
      ENDIF
      KSTP=0
      PRANG=0.0
      DO  9 I=1,K
 9       PRANG=PRANG+ABS(B(I))
      PREDR=2*PRANG
      W=PREDR/2.
      GYBIE=GY-B(IE)
      IF(W.LE.(3.*SIGER)) W=3.0*SIGER
      IF(W.LE.ABS(.05*GY)) W=ABS(0.05*GY)
      IF(SMBSQ.LE.0.0) GO TO 6
      STEP=W/SMBSQ
      DM=(W+GYBIE)/SMBSQ
      IF(KSEC.GT.0) GO TO 36
      DMS=FLOAT(LN-K)
      DMS=(AMLF1+AMLF2*(DMS-1.0))/DMS
      DMS=AMAX1(DMS,PEMS)
      IF(DMS.GT.0.0) GO TO 4
      PRANG=999999.
      GO TO 6
 4    PRANG=PRANG * SQRT(FLN/(DMS*REAL(K)))
 6    IF(PEMS.GT.0.0) GO TO 8
      FIT=999999.
      GO TO 11
 8    FIT=AMAX1(AMLF1,AMLF2)/PEMS
 11   WRITE(51,12) FIT
 12   FORMAT(5X,'LACK OF FIT RATIO=',F16.6)
      PSTOP=0.0
C
      IF(PRANG.LE.2.0) GO TO 16
      PRINT*,'*** EXPLORING THE PATH !!! ***'
      WRITE(51,13)
 13   FORMAT(T10,'****  EXPLORE THE PATH  ****')
      GO TO 36
C
C***  PREDICTED RANGE RATIO IS TOO SMALL - CHECK LACK OF FIT
C
 16   IF(FIT.LE.5.0) GO TO 24
      PRINT*, '***PRED. RANGE SMALL AND LACK OF FIT LARGE  ***'
```

```
      WRITE(51,14)
   14 FORMAT(5X,'***PRED. RANGE SMALL AND LACK OF FIT LARGE  ***')
      IF(RSQ.LT.0.90) GO TO 32
      PRINT*,'*** RSQ TOO LARGE TERMINATE THE SEARCH ***'
      WRITE(51,15)
   15 FORMAT(5X,' *** R-SQUARED IS TOO LARGE ***')
      CALL DONE(6)
   24 IF(NRUN.NE.LN+1) GO TO 28
      PRINT*,'***PRED. RANGE SMALL, LACK OF FIT SMALL ***'
      PRINT*,'***    RANDOM ERROR IS TO LARGE       ***'
      PRINT*,'*** INCREASE # OF ITERATIONS AND/OR    ***'
      PRINT*,'*** INCREASE STEP SIZES               ***'
      WRITE(51,17)
   17 FORMAT(T10,'*** PRED. RANGE SMALL, LACK OF FIT SMALL ***'/
     &T10,'***    RANDOM ERROR IS TO LARGE       ***'/
     &T10,'*** INCREASE # OF ITERATIONS AND/OR    ***'/
     &T10,'*** INCREASE STEP SIZES               ***')
      CALL DONE(2)
   28 PRINT*,'*** PRED. RANGE SMALL, LACK OF FIT SMALL ***'
      PRINT*,'*** HAVE REACHED A PLATEAU            ***'
      PRINT*,'*** ENTER SECOND ORDER PHASE          ***'
      WRITE(51,19)
   19 FORMAT(T10,'*** PRED. RANGE SMALL, LACK OF FIT SMALL ***'/
     & T10,'*** HAVE REACHED A PLATEAU            ***'/
     & T10,'*** ENTER SECOND ORDER PHASE          ***')
   32 PRINT*,'ENTER SECOND PHASE'
      NDES=2
      CALL ORDER
      RETURN
   36 DO 138 I=1,K
         BSTAR(I)=B(I)
         GG(I)=B(I)
  138    RR(I)=0
  140 CALL STEP5
      NGO=1
  142 IF(JGAM.NE.0) GO TO 146
      WRITE(51,144)
  144 FORMAT(T10,'CONSTRAINTS LIE IN DIRECTION OF PATH')
      GO TO 150
  146 WRITE(51,148)JGAM
  148 FORMAT('GOING TOWARDS CONSTARINT NO:',I4)
  150 IF(R.GE.1.5) GO TO 152
      NSS=1
      GO TO 162
  152 R=R-1.0
C
```

```
C*** FOLLOW THE PATH
C
    DO 40 I=1,K
 40     RR(I)=RR(I)+DM*BSTAR(I)
    DM=STEP
    DO 45 I=1,K
      J=INDX(I)
 45     X(J)=RR(I)*DELT(J)+CX(J)
 158 CALL SIMUL
C
C*** ENOUGH RUNS REMAIN?
C
    IF(NVAL.LT.NNOW) GO TO 60
    IRSRT=2
    CALL RESTAR(2)
 60  KSTP=KSTP+1
    IF(MRTF.EQ.1) KSTP=0
    IF(KSTP.GE.2) GO TO 80
    GO TO (142,170)NGO
 162 DEN=R*DM
    IF(DM.NE.STEP) DEN=GAMMIN
    DO 164 I=1,K
 164    RR(I)=RR(I)+DEN*BSTAR(I)
    DEN=0.0
    IA=0
    DO 166 I=1,K
      J=INDX(I)
      X(J)=RR(I)*DELT(J)+CX(J)
      DENOM=X(J)-HX(J)
      DEN=0.15*DELT(J)
      GAMAA=ABS(DENOM)
      IF(GAMAA.LE.DEN) GO TO 166
      IA=1
 166 CONTINUE
    NGO=2
    IF(IA.EQ.1) GO TO 158
    CALL DONE(2)
 170 CALL STP11A(EGSUM)
    IF(EGSUM.GT.(AMAX1(0.01,(1.5*SIGB**2)))) GO TO 172
    CALL DONE (7)
 172 DO 174 J=1,K
 174    BSTAR(J)=EE(J)
    STEP=STEP*SMBSQ
    SMBSQ=0
    DO 178 I=1,K
 178    SMBSQ=SMBSQ+BSTAR(I)**2
```

```
      STEP=STEP/SMBSQ
      GO TO 140

   80 IF(KSEC.EQ.0) GO TO 82
      CALL DONE(2)
   82 WRITE(51,83)
   83 FORMAT(10X,'*** SEEK NEW PATH ***')
      CALL ORDER
      RETURN
      END
      SUBROUTINE SUBSEC
$INCLUDE:'RSMCOM.FI'
      IF((NRRN.EQ.LN+2).AND.(NRUN.GT.LN+3))NDES=2
      IF(NDES.GE.2) GO TO 52
      J16=0
   10 KEEP=-10
      MTEN=-10
      IF (KEEP.EQ.MTEN) KEEP=K
      PRINT *,'*** # OF b(i)s > 2*SIGMAB=', KEEP
      WRITE(51,80)KEEP
   80 FORMAT(T10,'*** NUMBER OF B(I)s > 2*SIGMAB IS=',I5)

      IF (KEEP.LE.1) GO TO 14
      DO 6 J2=2,KEEP
         J2M=J2-1
         IF(B(J2).EQ.0.0) GO TO 8
         RATIO=(B(J2M)/B(J2))**2
         IF(RATIO.GT.20000.0) GO TO 8
    6 CONTINUE
      J2M=KEEP
      GO TO 12
C   8 WRITE(51,88)J2M

   12 KEEP=J2M
   14 TYSQ=0.0
C
C*** USE BEST POINT FOR THE CENTER OF NEXT FRACTIONAL FACTORIAL
      NLEFT=N-NRUN
      IF((NLEFT.GE.LN+2).AND.(KEEP.EQ.K)) GO TO 32
      IF(NLEFT.LT.4) GO TO 48
      NGM=NLEFT-2
      IF(NGM.GE.LN) GO TO 20
      J=1
      DO 16 I=1,LGSTR
         J=J+J
         IF(NGM.LT.J) GO TO 18
```

```
 16  CONTINUE
     STOP
 18  IR=I-1
     LN=J/2
 20  MAXK=LN-1
     IF(KEEP.GT.MAXK) THEN
C        WRITE(51,89) KEEP,MAXK
        KEEP=MAXK
     ENDIF
     K=KEEP
     J=2
     DO 24 I2=2,LGSTR
        J=J+J
        IF(KEEP.LT.J) GO TO 26
 24  CONTINUE
     STOP
 26  LN2=J
     IF(LN2.GE.LN) GO TO 28
     LN=LN2
     IR=I2
     IF(NLEFT.LT.16) GO TO 28
     IF(K.EQ.3) IR=3
     IF (K.EQ.3) LN=8
 28  WRITE(51,90)K
 90  FORMAT(T10,'*** K IS NOW ',I3,' ***')
     FLIE=K+1
     CALL WRITER(4)
C
C***  NEW CENTER POINT AND NEW DELTA'S
C
 32  DO 34 I=1,KT
        IBNDX(I)=0
        CX(I)=HX(I)
 34  CONTINUE
     DO 40 J=1,K
        I=INDX(J)
        IBNDX(I)=J
        IF(J16.GT.0) GO TO 40
        IF(B(J).NE.0.0) GO TO 36
        BLO=DELT(I)
        GO TO 38
 36     BLO=DELT(I)*0.5*ABS(B(1)/B(J))
        IF(BLO.GT.DELT(I)) BLO=DELT(I)
 38     DELT(I)=BLO
 40  CONTINUE
     CALL WRITER(3)
```

```
      YCENT=GY
      YSAVE=YCENT
      CALL SHIFT(J16)
      IF(J16.EQ.0) GO TO 46
      IF(J16.EQ.1) GO TO 42
      CALL DONE (6)
C 42  WRITE(51,43)
      CALL ORDER
      GO TO 10
 46   TSS=GYSS
      TYSQ=GY**2
      SIGSAV=SIGGY
      IRSRT=0
      GO TO 52
 48   PRINT*,'LESS THAN 4 RUNS LEFT'
      CALL DONE(1)
 52   RETURN
      END
```

```
C*********************************************************************
C*   THIS SUBROUTINE FITS A SECOND ORDER DESIGN                      *
C*********************************************************************
      SUBROUTINE SECOND
$INCLUDE:'RSMCOM.FI'
      IF(IRSRT.EQ.3) GO TO 36
      PRINT*,'*** SECOND ORDER DESIGN PHASE ***'
      WRITE(51,10)
 10   FORMAT(10X,'*** SECOND ORDER DESIGN PHASE ***')
      WRITE(51,11)
 11   FORMAT(1X,40('*'))
      DO 6 I=1,MAXK
        INDX(I)=JNDX(I)
        J=INDX(I)
 6      B(I)=OLDB(I)
      FLN=LN
      NREM=N-NRUN
      KSTP=0
      LFRST=1
      KSEC=(NREM-2)/2
      IF(KSEC.LT.1) KSEC=1
      IF(KSEC.GT.IR) KSEC=IR
      PRINT*,'*** # OF FACTORS IN 2ND PHASE= ',KSEC
      WRITE(*,*) (JNDX(I),I=1,KSEC)
      WRITE(51,7)KSEC
 7    FORMAT(1X,T10,'NUMBER OF FACRORS ARE ',I5)
      WRITE(51,8)(JNDX(I),I=1,KSEC)
 8    FORMAT(T10,'THE FACTORS ARE ',6I3)
      PRINT*,'*** AXIAL POINTS OF THE DESIGN ***'
      WRITE(51,9)
 9    FORMAT(10X,'****   AXIAL POINTS OF THE DESIGN  ****')
C
C***  FORM (X'Y) VECTOR
      KMIX=KSEC*(KSEC-1)/2
      KSM=KSEC+KMIX
      ICNST=KSM+KSEC+1
      IF(KMIX.EQ.0) GO TO 16
C
C***  MIXED TERMS
C
      DO 14 I=1,KMIX
        KSI=KSEC+I
        IRI=IR+I
 14     XPRMY(KSI)=FLN*B(IRI)
 16   SECTRM=(FLN+1.)*B(IE)-YCENT
C
```

```
C***  SECOND ORDER TERMS
C
      DO 18 I=1,KSEC
        KSMI=KSM+I
        XPRMY(KSMI)=SECTRM
 18       XPRMY(I)=FLN*B(I)
      XPRMY(ICNST)=(FLN+1.)*B(IE)
      FKSEC=KSEC
      STEP=0.5*SQRT(FKSEC)
      JSEC=1
      ALPSQ=SQRT(2.**KSEC)
      ALPHA=SQRT(ALPSQ)
      ALPHAB=ALPHA
 20   SWIT=-1.0
C
C*** NEW PART OF X'Y
C
      DO 22 I=1,K
        JKL=JNDX(I)
 22       X(JKL)=CX(JKL)
 24   JJ=JNDX(JSEC)
      X(JJ)=CX(JJ)+SWIT*DELT(JJ)*ALPHA
      IF(LFRST.EQ.0) GO TO 34
      DO 132 I=1,NCST
        CALL CKCST(I,VLATE,0)
        IF(VLATE.GE.(-0.0001)) GO TO 132
        DO 126 JK=1,6
          IF(KVIDN(I,JK).EQ.0) GO TO 126
          KVV=KVIDN(I,JK)
          IF(KVV.NE.JJ) GO TO 126
          GO TO 128
 126      CONTINUE
        GO TO 132
 128      ALPSTR=ABS((-VLATE+SWIT*DELT(JJ)*CCOEF(I,JK)*ALPHA)/
     & (DELT(JJ)*CCOEF(I,JK)))
        WRITE(51,129)
 129      FORMAT(T10,'CONSTRAINT VILOATION IN 2ND PHASE'/
     &T10,'CAHNGE VALUE OF ALPHA')
        IF(ALPSTR.LT.ALPHAB) ALPHAB=ALPSTR
 132  CONTINUE
      GO TO 38
C
C***  OBTAIN SIMUALTION RESPONSES FOR THE AXIAL POINTS
C
 34   CALL SIMUL
C
```

```
C***  ENOUGH RUNS REMAIN?
C
      IF(NVAL.LT.NNOW) GO TO 36
      IRSRT=3
      CALL RESTAR(2)
 36   XPRMY(JSEC)= XPRMY(JSEC)+Y*SWIT*ALPHA
      KSMJ=KSM+JSEC
      XPRMY(KSMJ)=XPRMY(KSMJ)+Y*SQRT(2.0**KSEC)
      XPRMY(ICNST)=XPRMY(ICNST)+Y
 38   IF(SWIT.GT.0.0) GO TO 40
      SWIT=-SWIT
      GO TO 24
 40   JSEC=JSEC+1
      IF(JSEC.LE.KSEC) GO TO 20
      IF(LFRST.EQ.0) GO TO 42
      LFRST=0
      ALPHA=ALPHAB
      ALPSQ=ALPHA**2
      JSEC=1
      GO TO 20
      PRINT*,'***SECOND ORDER DESIGN PHASE COMPLETED ***'
 42   WRITE(51,45)
 45   FORMAT(1X,T10,'*** SECOND ORDER DESIGN PHASE COMPLETED ***')
C
C***  GET QUADRATIC COEFFICIENTS
      CALL QUADR
      WRITE(51,46)
 46   FORMAT (10X,'ESTIMATED FIRST ORDER,MIXED,SECOND ORDER, AND',
     &/t11,' CONSTANT COEFFICIENTS ARE:')
      WRITE(51,47)(XXIXY(I),I=1,ICNST)
 47   FORMAT (3X,8(F10.5,2X))
C
C***  GET B MATRIX AND ITS LARGEST EIGEN VALUE
C
      CALL BMATRX
      RETURN
      END
```

```
C*********************************************************************
C*   THIS SUBROUTINE DETERMINES THE COEFFICIENTS FOR A               *
 C*   QUADRATIC FIT                                                  *
C*********************************************************************
      SUBROUTINE QUADR
$INCLUDE:'RSMCOM.FI'
      SECB=FLN+2.*ALPSQ
      SECC=FLN+2.*(ALPSQ**2)
      SECD=FLN
      SECN=FLN+2.0*FKSEC+1.0
      FKM1=FKSEC-1.0
      CKD=SECC+FKM1*SECD
      CMD=SECC-SECD
      BSQR=SECB**2
      SECA=CMD*(SECN*CKD-FKSEC*BSQR)
      SECP=CMD*CKD/SECA
      SECQ=-CMD*SECB/SECA
      SECR=(SECN*(CKD-SECD)-FKM1*BSQR)/SECA
      SECS=(BSQR-SECN*SECD)/SECA
C
C***  STORE COEEFICIENTS IN XXIXY ARRAY
C
      SECSUM=0.0
      DO 10 I=1,KSEC
        KSMI=KSM+I
 10     SECSUM=SECSUM+XPRMY(KSMI)
C
C***  B(0)
C
      XXIXY(ICNST)=SECP*XPRMY(ICNST)+SECQ*SECSUM
      DO 20 I=1,KSEC
        XXIXY(I)=XPRMY(I)/SECB
        KSMI=KSM+I
 20     XXIXY(KSMI)=SECS*SECSUM+(SECR-SECS)*XPRMY(KSMI)+
     &            SECQ*XPRMY(ICNST)
      IF (KMIX.EQ.0) GO TO 40
      DO 30 I=1,KMIX
        KSMI=KSEC+I
 30     XXIXY(KSMI)=XPRMY(KSMI)/SECD                    MIXED TERMS
 40   RETURN
      END
```

```
C*********************************************************************
C*    THIS SUBROUTINE FORMS THE B MATIX AND FINDS ITS                *
C*    LARGEST EIGEN VALUE                                            *
C*********************************************************************
      SUBROUTINE BMATRX
$INCLUDE:'RSMCOM.FI'
C
C*** FORM B MATRIX
C
      JJ=0
      DO 10 I=1,KSEC
        IOF=KSEC*(I-1)+I
        KSMI=KSM+I
        BMATG(IOF)=XXIXY(KSMI)
        IP=I+1
        IF(IP.GT.KSEC) GO TO 20
        DO 10 J=IP,KSEC
          JJ=JJ+1
          KJJ=KSEC+JJ
          DFFD=.5*XXIXY(KJJ)
          IOF=KSEC*(I-1)+J
          BMATG(IOF)=DFFD
          IOF=KSEC*(J-1)+1
          BMATG(IOF)=DFFD
 10   CONTINUE
 20   KSECSQ=KSEC**2
C
C*** FORM UPPER TRIANGULAR PART OF THE B MATRIX
C
      DO 40 I=1,KSEC
        DO 40 J=1,KSEC
          CALL LOCATE(I,J,IZ,KSEC,KSEC,1)
          CALL LOCATE(I,J,JJ,KSEC,KSEC,0)
 40         BMAT(IZ)=BMATG(JJ)
      DO 50 I=1,KSECSQ
 50     BTHEI(I)= BMATG(I)
C
C***  FIND THE EIGEN VALUES
C
      CALL EIGEN(BMAT,KSEC,1)
C
C***  LARGEST EIGEN VALUE STORED IN BMAT(1)
C
      THEMIN=BMAT(1)
C
C*** ALL EIGEN VALUES < 0 , IT IS MAXIMUM
```

```
      IF (THEMIN.LT.0.0) CALL CENTER
      RETURN
      END



C*********************************************************************
C*    THIS SUBROUTINE COMPUTES EIGEN VALUES OF A MATRIX              *
C*********************************************************************
      SUBROUTINE EIGEN(A,N,MV)
      DIMENSION A(10),R(1)
    5 RANGE=1.0E-6
      IF(MV-1) 10,25,10
   10 IQ=-N
      DO 20 J=1,N
        IQ=IQ+N
        DO 20 I=1,N
          IJ=IQ+I
          R(IJ)=0.0
          IF(I-J) 20,15,20
   15       R(IJ)=1.
   20 CONTINUE
   25 ANORM=0.0
      DO 35 I=1,N
        DO 35 J=I,N
          IF(I-J) 30,35,30
   30       IA=I+(J*J-J)/2
          ANORM=ANORM+A(IA)*A(IA)
   35 CONTINUE
      IF(ANORM) 165,165,40
   40 ANORM=1.414*SQRT(ANORM)
      ANRMX=ANORM*RANGE/REAL(N)
      IND=0
      THR=ANORM
   45 THR=THR/REAL(N)
   50 L=1
   55 M=L+1
   60 MQ=(M*M-M)/2
      LQ=(L*L-L)/2
      LM=L+MQ
   62 IF(ABS(A(LM))-THR)130,65,65
   65 IND=1
      LL=L+LQ
```

```
      MM=M+MQ
      X=0.5*(A(LL)-A(MM))
   68  Y= -A(LM)/SQRT(A(LM)**2+X**2)
      IF(X) 70,75,75
   70  Y=-Y
   75  SINX=Y/SQRT(2.*(1.+(SQRT(1.-Y*Y))))
      SINX2=SINX*SINX
   78  COSX=SQRT(1.-SINX2)
      COSX2=COSX*COSX
      SINCS=SINX*COSX
      ILQ=N*(L-1)
      IMQ=N*(M-1)
      DO 125 I=1,N
        IQ=(I*I-I)/2
        IF(I-L) 80,115,80
   80     IF(I-M) 85,115,90
   85     IM=I+MQ
        GO TO 95
   90     IM=IQ+M
   95     IF(I-L) 100,105,105
  100     IL=I+LQ
        GO TO 110
  105     IL=L+IQ
  110     X=A(IL)*COSX-A(IM)*SINX
        A(IM)=A(IL)*SINX+A(IM)*COSX
        A(IL)=X
  115     IF(MV-1) 120,125,120
  120     ILR=ILQ+I
        IMR=IMQ+I
        X=R(ILR)*COSX-R(IMR)*SINX
        R(IMR)=R(ILR)*SINX+R(IMR)*COSX
        R(ILR)=X
  125  CONTINUE
      X=2.0*A(LM)*SINCS
      Y=A(LL)*COSX2+A(MM)*SINX2-X
      X=A(LL)*SINX2+A(MM)*COSX2+X
      A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)
      A(LL)=Y
      A(MM)=X
  130  IF(M-N) 135,140,135
  135  M=M+1
      GO TO 60
  140  IF(L-(N-1)) 145,150,145
  145  L=L+1
      GO TO 55
  150  IF(IND-1) 160,155,160
```

```
155  IND=0
     GO TO 50
160  IF(THR-ANRMX) 165,165,45
165  IQ=-N
     DO 185 I=1,N
       IQ=IQ+N
       LL=I+(I*I-I)/2
       JQ=N*(I-2)
       DO 185 J=I,N
         JQ=JQ+N
         MM=J+(J*J-J)/2
         IF(A(LL)-A(MM)) 170,185,185
170        X=A(LL)
         A(LL)=A(MM)
         A(MM)=X
         IF(MV-1) 175,185,175
175        DO 180 K=1,N
             ILR=IQ+K
             IMR=JQ+K
             X=R(ILR)
             R(ILR)=R(IMR)
180            R(IMR)=X
185  CONTINUE
     RETURN
     END
C*********************************************************************
C*  THIS SUBROUTINE DETERMINES THE STATIONARY POINT X0               *
C*********************************************************************
     SUBROUTINE CENTER
$INCLUDE:'RSMCOM.FI'
     CDFLG=0.0
     RADCNT=0.0
     CALL MINV(BTHEI,KSEC,DET)
     IF(DET.EQ.0.0) THEN
       CALL DONE(4)
     ENDIF
     CALL MPRD(BTHEI,XXIXY,CDX,KSEC,KSEC,0,0,1)
     DO 10 I=1,KSEC
       BMAT(I)=0.0
       CDX(I)=-.5*CDX(I)
       BMAT(I)=CDX(I)
       RADCNT=RADCNT+CDX(I)**2
       CDXI=CDX(I)
       IF(CDXI.GE.(-1.).AND.CDXI.LT.1.0) GO TO 10
       CDFLG=1.0
 10  CONTINUE
```

```
      RADCNT=SQRT(RADCNT)
      IF(CDFLG.EQ.1.0) GO TO 20
C
C***  OBTAIN SIMULATION RESPONSE
C
      CALL CSTOP
 20   RETURN
      END

C*********************************************************************
C*  THIS SUBROUTINE OBTAINS THE SIMULATION RESPONSE FOR THE    *
C*  STATIONARY POINT AND THEN STOPS                            *
C*********************************************************************
      SUBROUTINE CSTOP
$INCLUDE:'RSMCOM.FI'
      BETAL=1.
 102  DO 10 I=1,KSEC
        JJ=JNDX(I)
 10       X(JJ)=DELT(JJ)*BETAL*CDX(I)+CX(JJ)
      IF(BETAL.NE.1) GO TO 12
      DO 18 I=1,NCST
        CALL CKCST(I,VLATE,0)
        IF(VLATE.GE.(-0.0001)) GO TO 18
        BETA=0
        DO 16 J=1,6
          KVV=KVIDN(I,J)
          IF(KVV.EQ.0) GO TO 16
          KVV=IBNDX(KVV)
          IF(KVV.EQ.0) GO TO 16
          BETA=BETA+ACOEF(I,J)*CDX(KVV)
 16       CONTINUE
        IF(BETA.EQ.0) GO TO 18
        BETA=-AZERO(I)/BETA
        IF(BETA.LT.BETAL) BETAL=BETA
 18   CONTINUE
      IF (BETAL.EQ.1.0) GO TO 12
      WRITE(51,13)
 13   FORMAT(T10,'PREDICTED CENTER IS IN CONSTRAINT REGION')
      GO TO 102
 12   CALL SIMUL
      CALL DONE(2)
      RETURN
      END
```

```
      SUBROUTINE RIDGE
$INCLUDE:'RSMCOM.FI'
      COMMON /PRG/XX(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
      IF(IRSRT.LT.4) GO TO 10
      IF(IRSRT.EQ.5) GO TO 46
      GO TO 20
 10   PRINT*,'***** RIDGE TO BE CLIMBED ***'
      WRITE(51,90)
 90   FORMAT(T10,50('*')/T10,'*** RIDGE TO BE CLIMBED ***')
      IRID=1
      RETURN
      EIGE=THEMIN
      ADDUP=2.*ABS(THEMIN)
      IF(ADDUP.EQ.0) ADDUP=1.0
      THEMAX=ADDUP
      RADFLG=0.0
      KSECP=KSEC+1
      END

C*********************************************************************
C*  THIS SUBROUTINE PRINTS THE RESULS                                *
C*********************************************************************
      SUBROUTINE DONE(ID)
      COMMON /PRG/XX(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
$INCLUDE:'RSMCOM.FI'
      PRINT *,'************ TERMINATE THE SEARCH **************'
      WRITE(51,10)
 10   FORMAT(T10,'++++++++++++ TERMINATE THE SEARCH ++++++++++++')
      GO TO (20,30,40,50,60,70,80) ID
 20   WRITE(51,22)
 22   FORMAT(T10,'+++ ALL AVAILABLE RUNS USED +++')
      PRINT*,'+++ ALL AVAILABLE RUNS USED +++'
      ITERM=1
      GO TO 90
 30   WRITE(51,33)
 33   FORMAT(T10,'+++ NOT ALL RUNS USED ---THIS IS THE BEST RESPONSE
     & THE PROGRAM CAN FIND+++')
      ITERM=1
      PRINT *,'+++NOT ALL RUNS USED ---THIS IS THE BEST RESPONSE
     & THE PROGRAM CAN FIND+++'
      GO TO 90
 40   WRITE(51,44)
 44   FORMAT(T10,'+++ THERE ARE NOT ENOUGH RUNS TO COMPLETE
INITIAL
     & DESIGN +++')
      PRINT *,'+++ THERE ARE NOT ENOUGH RUNS TO COMPLETE INITIAL
```

```
     & DESIGN +++'
      ITERM=2
      GO TO 100
   50 WRITE(51,55)
   55 FORMAT(T10,'+++ERROR IN MATRIX INVERSION - 2ND PHASE +++')
      PRINT *,'+++ERROR IN MATRIX INVERSION - 2ND PHASE +++'
      GO TO 90
   60 WRITE(51,66)
   66 FORMAT(T10,'+++ CONS. VIOLATION +++')
      GO TO 100
   70 WRITE(51,77)
   77 FORMAT(T10,'+++ INEFFICIENT TO INVEST MORE RUNS +++')
      GO TO 90
   80 CONTINUE
   90 IF(MIN.NE.0) GY=-GY
      WRITE(51,99)NRUN,M,GY,NTH
   99 FORMAT(//T10,'TOTAL NUMBER OF RUNS =',I5,/T10,'NUMBER OF
     &SIMULATION ITERATIONS =',I4,/T10,'OPTIMUM OBSERVED RESPONSE
     &=',F16.6,1X,'FOR ',I4,'TH RUN')
      WRITE(51,98)
   98 FORMAT( /T10,'VALUES OF X(1),.....X(K) ARE')
      DO 91 I=1,KT
        WRITE(51,92) I,HX(I)
   92   FORMAT(T15,'X(',I1,') =',F16.6)
   91 CONTINUE
  100 STOP
      END


     SUBROUTINE CODEX
  $INCLUDE:'RSMCOM.FI'
      COMMON /RSM4/IW1(4),IW2(4)
      DO 10 I=1,KSECSQ
   10   BTINV(I)=BMATG(I)
      DO 20 I=1,KSECSQ,KSECP
   20   BTINV(I)=BMATG(I)-THETA
      CALL MINV(BTINV,KSEC,DET)
      IF(DET.NE.0.0) GO TO 30
      CALL DONE(4)
   30 CALL MPRD(BTINV,XXIXY,CDX,KSEC,KSEC,0,0,1)
      DO 40 I=1,KSEC
   40   CDX(I)=-0.5*CDX(I)
      RETURN
      END
      SUBROUTINE ITER(IRET,IIN)
       COMMON /PRG/XX(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
```

```
$INCLUDE:'RSMCOM.FI'
    IF(IIN.EQ.1) GO TO 10
    BAL=YHAT
    AL=BETTR
    CAL=BETTR2
    GO TO 20
 10  BAL=RAD
    AL=RADL
    CAL=RADH
 20  IF(BAL.GT.AL) GO TO 30
    THEMAX=THETA
    THEM=THEMIN
    GO TO 40
 30  IF(BAL.LT.CAL) GO TO 50
    THEMIN=THETA
    THEM=THEMAX
 40  THETA=0.5*(THETA+THEM)
    IRET=1
    GO TO 60
 50  IRET=0
 60  RETURN
    END

C*********************************************************************
C*    THIS SUBROUTINE INVERTS A MATRIX                               *
C*    (from IBM's scientific subroutine with minor changes           *
C*********************************************************************
    SUBROUTINE MINV(A,N,D)
    DIMENSION A(16)
    COMMON /RSM4/L(4),M(4)
    D=1.
    NK=-N
    DO 80 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK=NK+K
      BIGA=A(KK)
      DO 20 J=K,N
        IZ=N*(J-1)
        DO 20 I=K,N
          IJ=IZ+I
          IF(ABS(BIGA)-ABS(A(IJ))) 15,20,20
 15         BIGA=A(IJ)
          L(K)=I
          M(K)=J
```

```
20     CONTINUE
       J=L(K)
       IF((J-K))35,35,25
25     KI=K-N
       DO 30 I=1,N
         KI=KI+N
         HOLD=-A(KI)
         JI=KI-K+J
         A(KI)=A(JI)
30       A(JI)=HOLD
35     I=M(K)
       IF(I-K) 45,45,38
38     JP=N*(I-1)
       DO 40 J=1,N
         JK=NK+J
         JI=JP+J
         HOLD=-A(KJ)
         A(JK)=A(JI)
40       A(JI)=HOLD
45     IF(BIGA) 48,46,48
46     D=0.0
       RETURN
48     DO 55 I=1,N
         IF(I-K) 50,55,50
50       IK=NK+I
         A(IK)=A(IK)/(-BIGA)
55     CONTINUE
       DO 65 I=1,N
         IK=NK+I
         HOLD=A(IK)
         IJ=I-N
         DO 65 J=1,N
           IJ=IJ+N
           IF(I-K) 60,65,60
60           IF(J-K) 62,65,62
62           KJ=IJ-I+K
           A(IJ)=HOLD*A(KJ)+A(IJ)
65     CONTINUE
       KJ=K-N
       DO 75 J=1,N
         KJ=KJ+N
         IF(J-K) 70,75,70
70       A(KJ)=A(KJ)/BIGA
75     CONTINUE
       D=D*BIGA
       A(KK)=1./BIGA
```

```
80  CONTINUE
    K=N
100  K=K-1
    IF(K) 150,150,105
105  I=L(K)
    IF(I-K) 120,120,108
108  JQ=N*(K-1)
    JR=N*(I-1)
    DO 110  J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK)=-A(JI)
110     A(JI)=HOLD
120  J=M(K)
    IF(J-K) 100,100,125
125  KI=K-N
    DO 130 I=1,N
      KI=KI+N
      HOLD=A(KI)
      JI=KI-K+J
      A(KI)=-A(JI)
130     A(JI)=HOLD
    GO TO 100
150  RETURN
    END

    SUBROUTINE ORDER
$INCLUDE:'RSMCOM.FI'
       DO 10 I=1,MAXK
      JNDX(I)=INDX(I)
      OLDB(I)=B(I)
10      TEMP(I)=B(I)
    IF(K.LT.2) RETURN
    DO 20 I=1,K
      PMX=ABS(B(1))
      NPMX=1
      DO 30 J=2,K
        IF(ABS(B(J)).LT.PMX) GO TO 30
        PMX=ABS(B(J))
        NPMX=J
30      CONTINUE
      TEMP(I)=B(NPMX)
      B(NPMX)=0.0
      LX(I)=INDX(NPMX)
20   CONTINUE
```

```
      DO 50 I=1,K
        B(I)=TEMP(I)
        INDX(I)=LX(I)
   50 CONTINUE
      RETURN
      END

      SUBROUTINE LOCATE(I,J,IR,N,M,MS)
      IX=I
      JX=J
      IF(MS-1)10,20,30
   10 IRX=N*(JX-1)+IX
      GO TO 36
   20 IF(IX-JX)22,24,24
   22 IRX=IX+(JX*JX-JX)/2
      GO TO 36
   24 IRX=JX+(IX*IX-IX)/2
      GO TO 36
   30 IRX=0
      IF(IX-JX) 36,32,36
   32 IRX=IX
   36 IR=IRX
      RETURN
      END



C*********************************************************************
C*  THIS SUBROUTINE MULTIPLIES MATRIX A BY MATRIX B                  *
C*********************************************************************
      SUBROUTINE MPRD(A,B,R,N,M,MSA,MSB,L)
      DIMENSION A(16),B(15),R(4)
      MS=MSA*10+MSB
      IF(MS-22) 30,10,30
   10 DO 20 I=1,N
   20    R(I)=A(I)*B(I)
      RETURN
   30 IR=1
      DO 90 K=1,L
        DO 90 J=1,N
          R(IR)=0
          DO 80 I=1,M
            IF(MS) 40,60,40
   40         CALL LOCATE(J,I,IA,N,M,MSA)
            CALL LOCATE(I,K,IB,M,L,MSB)
```

```
          IF(IA) 50,80,50
50          IF(IB) 70,80,70
60          IA=N*(I-1)+J
          IB=M*(K-1)+I
70          R(IR)=B(IR)+A(IA)*B(IB)
80        CONTINUE
90      IR=IR+1
    RETURN
    END

    SUBROUTINE RESTAR(I)
    COMMON /RSM1/X(15),Y
    COMMON /RSM2/LGSTR,LGSTK,IRSRT
    COMMON /RSM3/ALIST(220),LIST(72)
    COMMON /RSMC1/CLIST(450),LISTC(215)
    COMMON /RSMC2/CLST(4),LSTC(4)
    OPEN (49,FILE='RSMIN.DAT',FORM='FORMATTED',STATUS='UNKNOWN')
    OPEN(50,FILE='RSMOUT,DAT',FORM='FORMATTED',STATUS=
   &'UNKNOWN')
    IF (I.EQ.1) THEN
     READ(49,16)ALIST,X,Y,CLIST,CLST
     READ(49,18)LIST,IRSRT,LISTC,LSTC
    ELSE
     WRITE(50,16)ALIST,X,Y,CLIST,CLST
     WRITE(50,18)LIST,IRSRT ,LISTC,LSTC
       STOP
    ENDIF
 16  FORMAT(5E15.8)
 18  FORMAT(16I5)
    END

C*********************************************************************
C*  THIS SUBROUTINE RETURNS THE SIMULATION RESPONSE                  *
C*  FROM M ITERATIONS FOR THE CURRENT FACTOR VALUES                  *
C*********************************************************************
    SUBROUTINE SIMUL
    COMMON /PRG/XX(10),UBEST,IRID,NVAR,CPU,MIN,ITERM,NREM
$INCLUDE:'RSMCOM.FI'
    NVAL=NVAL+1
    NRRN=NRRN+1
    NRUN=NRUN+1
    PRINT*, '**** RUN NUMBER :',NRUN
    WRITE(51,80) NRUN,M
 80  FORMAT(T10,60('*'),/T10, 'RUN NUMBER :',I4,2X,'NUMBER OF
   + ITERATIONS: ',I3)
C
```

```
C*** GET SIMULATION RESPONSE FOR M ITERATIONS
C
    YS=0
    SYSQ=0
    CSYSQ=0
    DO 10 I=1,M
      CALL OBJECT
      print*,'++++from SIMUL Y=',y
      PRINT*,'ITERATION',I,'Y=',Y
      WRITE(51,82) I,Y
 82     FORMAT(T15,'ITERATION=',I2,5X,'Y=',F16.6)
      IF(I.EQ.1) YFIRST=Y
      CSYSQ=CSYSQ+(Y-YFIRST)**2
      SYSQ=SYSQ+(Y*Y)
 10     YS=YS+Y
    Y=YS/FLM
    PRINT*,'AVG. RESPONSEIS =',Y
    WRITE(51,84)Y
 84  FORMAT(T10,'*** AVG. RESPONSE IS :',F15.6)
    TSS=TSS+SYSQ
    TYSQ=TYSQ+Y*Y
    IF(M.EQ.1) THEN
      SIGHI=0.0
      GO TO 12
    ENDIF
C
C*** CALCULATE VARIANCES
C
    YS=YS-FLM*YFIRST
    SIGHI=CSYSQ-(Y**2)/FLM
    SIGSAV=SIGSAV+SIGHI
 12  IF(MIN.NE.0) Y=-Y
    IF(NRUN.EQ.1) GO TO 14
C
C*** MRTF=1 IF BEST RESPONSE, MRTF=0 OTHERWISE
C
    MRTF=0
    IF(Y.LE.GY) GO TO 22
 14  MRTF=1
    GY=Y
    GYSS=SYSQ
C
C*** SET THE FLAG TO DETERMINE WHEN TO ENTER SECOND PHASE
C
    IF(NRRN.GT.LN) NDES=0
    SIGGY=SIGHI
```

```
      NTH=NRUN
      DO 16 I=1,KT
   16    HX(I)=X(I)
      WRITE(51,86)
   86  FORMAT(T40,'THIS IS THE OPTIMUM RESPONSE THUS FAR '
     */T40,37('=') )
   22  DO 18 I=1,KT
         WRITE(51,88) I,X(I)
         write(*,88) i,x(i)
   88     FORMAT(T10,'X(',I1,') =',F15.6)
   18  CONTINUE
      WRITE(51,90)
   90  FORMAT(T10,60('*'))
C
C*** CHECK REMAINING RUNS
C
      IF(NRUN.LT.N) GO TO 28
      CALL DONE(1)
   28  RETURN
      END

      SUBROUTINE CKCST(I,VLATE,LIT)
$INCLUDE:'RSMCOM.FI'
      VLATE=CZERO(I)
      DO 10 J=1,6
         IF(KVIDN(I,J).EQ.0) GO TO 14
         KVV=KVIDN(I,J)
         VLATE=VLATE+CCOEF(I,J)*X(KVV)
   10  CONTINUE
   14  IF((LIT.LT.0).AND.(VLATE.LT.(-.0001))) THEN
         DO 11 J=1,6
            IF(KVIDN(I,J).EQ.0) RETURN
            KVV=KVIDN(I,J)
            DELT(KVV)=0.0
            KVV=IBNDX(KVV)
   11        B(KVV)=0.0
      ENDIF
      RETURN
      END
      SUBROUTINE SHIFT(J16)
$INCLUDE:'RSMCOM.FI'
      J16=0
      KVIL=0
  100  CALL STEP2(LITE,1)
      IF(LITE.EQ.0) GO TO 114
      KVIL=KVIL+1
```

```
      IF(KVIL.GT.1) GO TO 106
      WRITE(51,10)
   10  FORMAT(T10,'CONSTRAINTS VILOATED-SHIFT THE DESIGN')
      DO 14 J=1,K
        I=INDX(J)
   14     CX(I)=2.0*HX(I)-S(I)
      CALL WRITER(1)
      GO TO 100
  106  DO 18 J=1,K
        I=INDX(J)
   18    CX(I)=HX(I)
      WRITE(51,19)
   19  FORMAT(T10,'VILOATIONS IN SHIFTED DESIGN, SHIFT BACK AND
     * INACTIVATE FACTORS')
      ISHFT=0
      CALL WRITER(1)
      CALL STEP2(LITE,-1)
      J16=1
      DO 112 J=1,K
         I=INDX(J)
         IF(DELT(I).GT.0) GO TO 114
  112  CONTINUE
      J16=-1
  114  RETURN
      END

      SUBROUTINE STEP2(LITE,LIT)
$INCLUDE:'RSMCOM.FI'
      DEE=0
      LITE=0
      DO 6 I=1,NCST
        KHIT(I)=0
        LFRST=0
        CALL FACTOR
        DO 6 I2=1,LN
          LFRST=I2
          CALL FACTOR
          CALL CKCST(I,VLATE,LIT)
          IF(VLATE.GE.(-0.0001)) GO TO 6
          LITE=1
          IF(LIT.LT.0) GO TO 6
          IF(LIT.LE.0) GO TO 6
          DEN=0
          DO 12 J=1,6
            JT=KVIDN(I,J)
            IF(JT.EQ.0) GO TO 12
```

```
            DEN=DEN+(CCOEF(I,J)*DELT(JT))**2
12          CONTINUE
          D=-VLATE/SQRT(DEN)
          IF(D.LE.DEE) GO TO 6
          DEE=D
          ISHFT=I2
          DO 14 I3=1,KT
14            S(I3)=X(I3)
 6  CONTINUE
    IF(LITE.GT.0) GO TO 11
    WRITE(51,8)
 8  FORMAT(T10,'ALL POINTS IN FRACT. FACT. SATISFY ALL C
   &CONSTRAINTS')
11  DO 22 I=1,KT
22      IBNDX(I)=0
    DO 24 I=1,K
       J=INDX(I)
24     IBNDX(J)=I
C
C***  CONVERT CONSTRAINTS TO THE CODED FACTORS
C
    DO 16 I=1,NCST
       AZERO(I)=CZERO(I)
       DO 16 J=1,6
         KVV=KVIDN(I,J)
         IF(KVV.EQ.0) GO TO 16
         AZERO(I)=AZERO(I)+CCOEF(I,J)*CX(KVV)
         ACOEF(I,J)=CCOEF(I,J)*DELT(KVV)
16  CONTINUE
    RETURN
    END
C
    SUBROUTINE STEP5
$INCLUDE:'RSMCOM.FI'
    JGAM=0
    GAMMIN=999999.0
    NTIE=0
    DO 104 I=1,NCST
       DENOM=0
       GAMAA=0
       GAMZZ=0
       DO 100 JZ=1,6
          JT=KVIDN(I,JZ)
          IF(JT.EQ.0) GO TO 100
          JT=IBNDX(JT)
          IF(JT.EQ.0) GO TO 100
```

```
          GAMAA=GAMAA+(ACOEF(I,JZ)*RR(JT))
          DENOM=DENOM+(ACOEF(I,JZ)*BSTAR(JT))
100     CONTINUE
        GAMZZ=-AZERO(I)-GAMAA
        IF(DENOM.EQ.0) THEN
          GAMZZ=-1.0
          GO TO 102
        ENDIF
        GAMZZ=GAMZZ/DENOM
102     IF(GAMZZ.LE.0)GO TO 104
        IF(GAMZZ.GE.GAMMIN) GO TO 104
        GAMMIN=GAMZZ
        JGAM=I
104     GAM(I)=GAMZZ
      DO 106 J=1,K
        RJ=RR(J)
        IF(RJ.NE.0.0) GO TO 108
106   CONTINUE
      GO TO 110
108   DM=STEP
C*** CALCULATE NUMBER OF STEPS
110   R=(GAMMIN-DM)/STEP+1.0
      IF(GAMMIN.EQ.999999.0) R=GAMMIN
      RETURN
      END

      SUBROUTINE STP11A(EGSUM)
$INCLUDE:'RSMCOM.FI'
      NTIE=0
      DO 100 I=1,K
100     EE(I)=BSTAR(I)
      DO 106 I=1,NCST
        IF(GAM(I).GT.1.05*GAMMIN) GO TO 104
        IF(GAM(I).LT.0.95*GAMMIN) GO TO 104
        NTIE=NTIE+1
        GAMAA=0
        DENOM=0
        DO 102 J=1,6
          JT=KVIDN(I,J)
          IF(JT.EQ.0) GO TO 102
          JT=IBNDX(JT)
          IF(JT.EQ.0) GO TO 102
          GAMAA=GAMAA+ACOEF(I,J)*GG(JT)
          DENOM=DENOM+(ACOEF(I,J)**2)
102     CONTINUE
        GAM(I)=ABS(GAMAA/SQRT(DENOM))
```

```
      GO TO 106
104     GAM(I)=0.0
106  CONTINUE
     DO 120 I2=1,NTIE
       SMALL=999999.0
       DO 108 JJ=1,NCST
         IF(GAM(JJ).GE.SMALL) GO TO 108
         IF(GAM(JJ).EQ.0) GO TO 108
         SMALL=GAM(JJ)
         ICON=JJ
108      CONTINUE
       GAM(ICON)=0.0
       WRITE(51,110)ICON
110      FORMAT(T10,'CONSTRAINT',I3,' HAS BEEN HIT')
       ABNUM=0
       AADEN=0
       DO 112 J=1,6
         JT=KVIDN(ICON,J)
         IF(JT.EQ.0) GO TO 114
         JT=IBNDX(JT)
         IF(JT.EQ.0) GO TO 112
         IF(EE(JT).EQ.0) GO TO 112
         ABNUM=ABNUM+ACOEF(ICON,J)*EE(JT)
         AADEN=AADEN+(ACOEF(ICON,J))**2
112      CONTINUE
114      IF(AADEN.GT.0) GO TO 116
       ABFCT=0
       GO TO 118
116      ABFCT=ABNUM/AADEN
118      CALL ST12A(ICON)
       KHIT(ICON)=I
120  CONTINUE
     EGSUM=0
     DO 122 J=1,K
122      EGSUM=EGSUM+EE(J)*GG(J)
     RETURN
     END

     SUBROUTINE ST12A(ICON)
$INCLUDE:'RSMCOM.FI'
     DO 100 J=1,6
       JT=KVIDN(ICON,J)
       IF(JT.EQ.0) GO TO 102
       JT=IBNDX(JT)
       IF(JT.EQ.0) GO TO 100
       EE(JT)=EE(JT)-ABFCT*ACOEF(ICON,J)
```

```
      IF(ABS(EE(JT)).LT.ABS(0.0001*GG(JT))) EE(JT)=0
100  CONTINUE
102  DO 108 I=1,NCST
      IF(KHIT(I).EQ.0) GO TO 108
      EASUM=0
      DO 104 J=1,6
         JT=KVIDN(I,J)
         IF(JT.EQ.0) GO TO 104
         JT=IBNDX(JT)
         IF(JT.EQ.0) GO TO 104
         EASUM=EASUM+EE(JT)*ACOEF(I,J)
104     CONTINUE
      IF(EASUM.GT.0) GO TO 108
      DO 106 J=1,6
         JT=KVIDN(I,J)
         IF(JT.EQ.0) GO TO 108
         JT=IBNDX(JT)
         IF(JT.EQ.0) GO TO 106
         IF(ACOEF(I,J).NE.0) EE(JT)=0
106     CONTINUE
108  CONTINUE
    RETURN
    END


C*************************************************************************
C*   THIS SUBROUTINE WRITES THE RESULTS TO THE SCREEN OR          *
C*   TO THE FILE OR BOTH                                          *
C*************************************************************************
    SUBROUTINE WRITER(INPFLG)
$INCLUDE:'RSMCOM.FI'
    OPEN (51,FILE='RESOUT.DAT',FORM='FORMATTED',STATUS=
   &'UNKNOWN')
    GOTO(10,20,10,40) INPFLG
 10  WRITE(*,11)
 11  FORMAT(1X,T10,'STARTING VALUE OF X(I)',T40,'DELTA VALUE FOR
   & X(I)')
    DO 12 J=1,K
       I=INDX(J)
       WRITE(*,14)I,CX(I),DELT(I)
 14     FORMAT(T4,'X(',I1,')',T15,F12.4,T45,F12.4)
 12  CONTINUE
    IF(INPFLG.EQ.3) GO TO 20
    RETURN
 20  WRITE(51,11)
    DO 15 J=1,K
```

```
      I=INDX(J)
      WRITE(51,14)I,CX(I),DELT(I)
15   CONTINUE
     RETURN
40   WRITE(51,41) (INDX(I),I=1,K)
41   FORMAT(20HACTIVE FACTORS ARE /(20X,20I3))
     RETURN
     END
```

```
C*************************************************************************
C*                    SLAM USER-WRITTEN SUBROUTINES                      *
C*************************************************************************
      SUBROUTINE INTLC
     COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,
     *MSTOP,NCLNR,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),
     *TNEXT,TNOW,XX(100)
     COMMON/SC1/IFLAG,NDB,BFLAG,FLG,BVAR,CILL,CIUL,YDBAR
     OPEN(40,FILE='SIN.DAT',ACCESS='DIRECT',RECL=20,FORM=
     *'FORMATTED')
     READ(40,100,REC=1)XX(12)
 100 FORMAT(F20.8)
     N=INT(XX(12))
     DO 10 I=1,N
       K=I+1
       READ(40,100,REC=K) XX(I)
  10 CONTINUE
     XX(1)=60./XX(1)
     XX(14)=0
     XX(15)=800
     XX(20)=100000
     NDB=0
     IFLAG=0
     BFLAG=1
     FLG=0
     RETURN
     END

     SUBROUTINE OTPUT
     COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,
     *MSTOP,NCLNR,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),
     *TNEXT,TNOW,XX(100)
     COMMON/SC1/IFLAG,NDB,BFLAG,FLG,BVAR,CILL,CIUL,YDBAR
     COMMON/SC2/JFLG
     COMMON/SYB/YB(400)ISTP,FLEN
     REAL Z(400)
     OPEN(41,FILE='SOUT.DAT',ACCESS='DIRECT',RECL=20,FORM=
     *'FORMATTED')
     OPEN(45,FILE='BATCH.DAT',ACCESS='APPEND',STATUS='UNKNOWN')
     OPEN(46,FILE='TTEST.DAT',ACCESS='DIRECT',RECL=20,FORM=
     *'FORMATTED',MODE='READWRITE')
     E=1.
     FIND=CCAVG(1)
     FSTD=CCSTD(1)
     TC=4.02/XX(1)+.4*30.*(CCAVG(1)/60.)
     WRITE(41,100,REC=1)TC
```

```
      WRITE(41,100,REC=2)FIND
      WRITE(41,100,REC=3)CCNUM(1)
      IF(BFLAG.EQ.1.OR.IFLAG.EQ.0) THEN
        WRITE(41,100,REC=4)E
      ENDIF
      IF(TNOW.GE.XX(20).AND.JFLG.EQ.1) PRINT*,'NOT ENOUGH RUN'
      IF(BFLAG.EQ.1)PRINT*,'BIAS STILL EXISTS,SIMULATION IS NOT
     * LONG ENOUGH! '
      IF(IFLAG.EQ.0)PRINT*,'DID NOT REACH STEADY STATE'
      WRITE(45,46)XX(1),BFLAG,IFLAG,TNOW,CCNUM(1),FIND,CILL,
     *CIUL,BVAR
   46 FORMAT(5X,'SERVICE RATE=',F5.2,3X,'BIAS=',F2.0,'STEADY
     * STATE=',I2,'SIMULATION TIME=',F10.2/T10,'NUMBER OF
     * OBSERVATIONS=',F10.0,'MEAN RESPONSE=',F10.2/
     *T10,'CI LOWER=',F10.2,3X,'CI UPPER=',F10.2,'VAR=',F10.3)
  100 FORMAT(F20.8)
      SUM=0.0
      READ(46,100,REC=1)PISTP
      IF(PISTP.LT.0) GO TO 200
      READ(46,100,REC=2)PFLEN
      ISTART=MAX(PISTP,ISTP)
      IFEND=MIN(PFLEN,FLEN)
      NUMOBS=IFEND-ISTART
      DO 10 I=ISTART,IFEND
        READ(46,100,REC=I)PY
        READ(42,100,REC=I)Y
        Z=Y-PY
        SUM=SUM+Z
   10 CONTINUE
      ZBAR=SUM/NUMOBS
      SUM=0
      DO 11 I=ISTART,IFEND
        READ(46,100,REC=I)PY
        READ(42,100,REC=I)Y
        SUM=SUM+((Y-PY)-ZBAR)**2
   11 CONTINUE
      VARIAN=SUM/(NUMOBS*(NUMOBS-1))
C
C***  CALCULATE 90% CONFIDENCE INTERVAL
C
      DELTA=1.645*SQRT(VARIAN)
      ZL=ZBAR-DELTA
      ZU=ZBAR+DELTA
      IF(ZU*ZL.LE.0) THEN
        READ(41,100,REC=5)REDATA
        REDATA=REDATA+1
```

```
      WRITE(41,100,REC=5)REDATA
       WRITE(45,*) 'REDUNDANT DATA SEARCH=',REDATA
    ENDIF
200  WRITE(46,100,REC=1)ISTP
    WRITE(46,100,REC=2)FLEN
    DO 12 I=1,FLEN
      READ(42,100,REC=I)Y
      WRITE(46,100,REC=I)Y
12   CONTINUE
    RETURN
    END


     SUBROUTINE EVENT(I)

    COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,
   *MSTOP,NCLNR
   *,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,
   * XX(100)
    COMMON/SC1/IFLAG,NDB,BFLAG,FLG,BVAR,CILL,CIUL,YDBAR
    COMMON/SC2/JFLG
     COMMON/SYB/YB(400)ISTP,FLEN
    INTEGER*4 K
    OPEN(42,FILE='SDATA.DAT',ACCESS='DIRECt',FORM='FORMATTED',
   *RECL=20)
     open(45,FILE='BATCH.DAT',ACCESS='APPEND',STATUS='UNKNOWN')
    IFLAG=0
    TRP=1000
    IF(TNOW.LE.1000) THEN
      XX(14)=0
      RETURN
    ENDIF
    IF(I.EQ.2) RETURN
    K=INT4(XX(14))
    WRITE(42,100,REC=K) XX(11)
100  FORMAT(F20.4)
    IFLAG=0
    IF((XX(14)).EQ.XX(15)) THEN
          GO TO 2
    ELSE
          IF (TNOW.GE.XX(20)) THEN
              PRINT*,'THE LENGTH OF SIMULATION IS NOT ENOUGH'
               JFLG=1
          ENDIF
          RETURN
    ENDIF
```

```
2     IF(BFLAG.EQ.1) THEN
      CALL BIAS(XX(15),BFLAG,NDB)
      IF(BFLAG.EQ.0) GO TO 10
      XX(15)=XX(15)+800
      RETURN
    ENDIF
10  iF(FLG.EQ.0) THEN
      PRINT*,'*******    NO  BIAS  *******'
      ISTP=(INT(XX(15)/800)-1)*800+NDB*40+1
      XX(15)=NDB*40+XX(15)
      PRINT*,'XX15 AFTER BIAS',XX(15)
      FLG=1
      PRINT*,'NDB=',NDB
      PRINT*,'-----DEL. POINT=',ISTP
      IF(NDB.GT.0)RETURN
      ENDIF
    PRINT*,'CALLING BATCH',XX(14)
    CALL BATCH (XX(15),istp)
    PRINT*,'BATCH FLAG',IFLAG
    IF(IFLAG.EQ.1) THEN
          MSTOP= -1
          FLEN=XX(15)
    ELSE
         xx(15)=xx(15)+400.
         PRINT*,'BATCH XX15=',XX(15)
    ENDIF
200 RETURN
    END
```

```
C*********************************************************************
C*   THIS SUBROUTINE CHECKS SIMULATION OUTPUT FOR INITAL             *
C*   BIAS  AND DETERMINES THE TRUNCATION POINT                       *
C*   BY SCHRUBEN'S METHOD                                            *
C*********************************************************************
    SUBROUTINE BIAS(XX7,BFLAG,NDB)
     REAL Y(40),ts(30)
     INTEGER BSIZE,DF
     data ts/6.314,2.92,2.358,2.132,2.015,
    &1.943,1.895,1.86,1.833,1.812,1.796,1.782,
    &1.771,1.761,1.753,1.746,1.74,1.734,1.729,
    &1.725,1.721,1.717,1.714,1.711,1.708,1.706,
    &1.703,1.701,1.699,1.697/
     DATA BSIZE/40/
     PRINT*,'INSIDE BIAS'
     N= INT(XX7)
     NB=N/BSIZE
     NHB=NB/2
     DF=NHB-1
     SUM=0
     DO 10 I=1,NB
 10       Y(I)=0
     DO 11 I=1,NB
        DO 12 K=(BSIZE*(I-1)+1),BSIZE*I
c         DO 12 K=1,BSIZE
          READ(42,100,REC=K) X
          READ(42,100)X
 100        FORMAT(F20.4)
 12         Y(I) = Y(I)+X
       Y(I) =Y(I)/BSIZE
 11   SUM=SUM+Y(I)
      YN=SUM/NB
      PRINT*,'GRAND AVG=',YN
C
C***   COMPUTE SAMPLE VARIANCE BY USING HALF OF THE DATA
C
      SUM=0
      DO 13 I=NHB+1,NB
 13       SUM=SUM+Y(I)
     YNT=SUM/NHB
     VARIAN=0
     DO 15 I=NHB+1,NB
      VARIAN= VARIAN+sqrt((Y(I)-YNT)**2)
 15   CONTINUE
     VARIAN=VARIAN*(NHB*BSIZE)/(NHB*(NHB-1))
     IF(VARIAN.EQ.0) THEN
```

```
      BFLAG=0
      RETURN
    ENDIF
    VAR=SQRT(VARIAN)
    PRINT*,'CALCULATE VARIANCE=',VAR
    TSTAT=TS(MIN(30,DF))
    NDB=0
C
C*** COMPUTE THE TEST STATISTICS
C
 50  NR=(NB-NDB)
    YN=0
    DO 16 I=1+NDB,NB
 16      YN=YN+Y(I)
    YN=YN/(NB-NDB)
    FTERM=SQRT(45.)/(VAR*REAL(NR**1.5))
    STERM=0
      DO 20 I=1+NDB,NB
      YK=0
      DO 21 K=1,I
 21        YK=YK+Y(K)
      YK=YK/I
 20     STERM= STERM+(1-I/NR)*I*(YN-YK)
    T = FTERM*STERM
     PRINT*,'TEST STATISTICS CALCULATED' ,T
C
C ***  CHECK THE HYPOTHESIS
C
    IF(ABS(T).LE.TSTAT) THEN
      BFLAG=0
      PRINT*,'NO INITAL BIAS EXIST'
   ELSE
      BFLAG=1
      PRINT*,'INITIAL BIAS STILL EXISTS'
      NDB=NDB+1
      IF(NDB.LE.NB) THEN
          GO TO 50
      ELSE
          PRINT*,'BIAS EXISTS, SIM. LENGTH NOT ENOUGH'
      ENDIF
    ENDIF
    RETURN
    END
```

```
      SUBROUTINE BATCH(XX4,istp)
C*********************************************************************
C *   THIS SUBROUTINE COMPUTES CONFIDENCE INTERVALS FOR              *
C *   STEADY STATE SIMULATION BY USING LAW & CARSON'S                *
C *   SEQUENTIAL PROCEDURE                                           *
C*********************************************************************
      COMMON/SC1/IFLAG,NDB,BFLAG,FLG,BVAR,CILL,CIUL,YDBAR
      COMMON/SYB/YB(400),ISTP,FLEN
      INTEGER  FN
      INTEGER*4 MI,L,II,J
      REAL TEMP(200)
      DATA N,F,U,T,GAMMA/40,10,0.4,2.023,0.075/
      OPEN(42,FILE='SDATA.DAT',STATUS='UNKNOWN',ACCESS='DIRECT',
     *FORM='FORMATTED',RECL=20)
      OPEN (44,FILE='CONFI.DAT',STATUS='UNKNOWN')
      OPEN(45,FILE='BATCH.DAT',STATUS='UNKNOWN',ACCESS='APPEND')
      IFLAG=0
C
C *** MI IS THE NUMBER OF OBSERVATIONS
C
      MI=INT4(XX4)-ISTP+1
C
C *** READ DATA AND COMPUTE YBAR'S ***
C
C
C *** INITIAL BATCH SIZE ***
C
 5    FN=F*N
      L = MI/FN
C
C ***  FIND MEAN OF EACH BATCH
C
      K=1
      DO 2  II=1,MI,L
       SUM =0
       DO 3 J=II,II+L-1
          READ(42,100,REC=(j+istp-1)) Y
 100        FORMAT(F20.4)
          SUM=SUM+Y
 3     CONTINUE
       YB(K)=SUM/L
       K=K+1
 2    CONTINUE
C
C *** FIND OVERALL MEAN
C
```

```
      YDBAR=GRMEAN(YB,1,FN)
      PC1=PLAG(YB,YDBAR,1,FN)
      YDF=GRMEAN(YB,1,FN/2)
      P11=PLAG(YB,YDF,1,FN/2)
      YDL=GRMEAN(YB,FN/2+1,FN)
      P12=PLAG(YB,YDL,FN/2+1,FN)
      PJACK=2*PC1-(P11+P12)/2
      POLD=PJACK
      BVAR=VAR(YDBAR,FN,YB)
      IF (PJACK.GE.U) THEN
        GO TO 200
      ELSE
        IF(PJACK.LE.0) THEN
          GO TO 180
        ELSE
          GO TO 150
        ENDIF
      ENDIF
C
C *** DOUBLE BATCH SIZE
C
 150  NB=F*N/2
      L=MI/NB
      K=1
      TEMP(1)=(YB(1)+YB(2))/2.
      DO 151 II=3,FN,2
        K=K+1
        TEMP(K)=(YB(II)+YB(II+1))/2
 151  CONTINUE
      PC1=PLAG(TEMP,YDBAR,1,NB)
      YDF=GRMEAN(TEMP,1,NB/2)
      P11=PLAG(TEMP,YDF,1,NB/2)
      YDL=GRMEAN(TEMP,NB/2+1,NB)
      P12=PLAG(TEMP,YDL,NB/2+1,NB)
      PJACK=2*PC1-(P11+P12)/2
      BVAR=VAR(YDBAR,NB,TEMP)
      IF (PJACK.GE.POLD) GO TO 200
 180  NB=N
      L=MI/NB
      J=1
      DO 185 II=1,N
        SUM=0
        DO 186 K=(II-1)*F+1,F*II
          SUM=SUM+YB(K)
 186    CONTINUE
        TEMP(J)=SUM/F
```

```
      J=J+1
 185  CONTINUE
C
C***  COMPUTE THE ESTIMATE OF THE VARIANCE
C
      SUM=0
      DO 190 J=1,N
         SUM =SUM+(TEMP(J)-YDBAR)**2
 190  CONTINUE
      SS=SUM/(N-1)
      BVAR=SQRT(SS/N)
      WRITE(45,*) 'VAR=',BVAR
      DELTA= T*BVAR
      IF (DELTA/YDBAR.LE.GAMMA) GO TO 500
C
C *** NOT ENOUGH PRECISION - COLLECT ADDITIONAL OBSERVATIONS
 200  RETURN
C
C *** REACHED THE STEADY STATE ***
C *** 90% CONFIDENCE INTERVAL IS GIVEN BY ***
C
 500  CILL = YDBAR-DELTA
      CIUL = YDBAR +DELTA
      PRINT*, 'LOWER=',CILL,'  UPPER=',CIUL
      WRITE(45,*)'LOWER=',CILL,'  UPPER=',CIUL,DELTA
      WRITE(45,*)'MEAN=',YDBAR
      IFLAG=1
      RETURN
      END


C
C *** THIS FUNCTION COMPUTES THE MEAN OF A SAMPLE
C

      FUNCTION GRMEAN(Y,I,J)
      REAL Y(*)
      SUM = 0
      DO 10 K=I,J
 10   SUM=SUM+Y(K)
      GRMEAN =SUM/(J-I+1)
      RETURN
      END
```

```
C
C*** THIS FUNCTION COMPUTES PI(L)- LAG i CORRELATION
C
      REAL FUNCTION PLAG(YB,YDBAR,I,N)
      REAL YB(*)
      SUM1=0
      SUM2=0
      DO 10 J=I,N-1
        SUM1=SUM1+(YB(J)-YDBAR)*(YB(J+1)-YDBAR)
        SUM2=SUM2+(YB(J)-YDBAR)**2
 10   CONTINUE
      SUM2=SUM2+(YB(N)-YDBAR)**2
      PLAG=SUM1/SUM2
      RETURN
      END

      REAL FUNCTION VAR(YDBAR,N,Y)
      REAL y(*)
      SUM=0
      DO 10 I=1,N
 10     SUM=SUM+(Y(I)-YDBAR)**2
      SS=SUM/(N-1)
      VAR=SQRT(SS/N)
      RETURN
      END
```

VITA 2

Zeynep Aysegul Karacal

Candidate for the Degree of

Doctor of Philosophy

Thesis : THE DEVELOPMENT OF AN AUTOMATED DISCRETE-EVENT SIMULATION OPTIMIZATION SYSTEM

Major Field : Industrial Engineering and Management

Biographical:

Personal Data: Born in Istanbul, Turkey, on March 1,1960, the daughter of R. Orhan and Guner Izgu. Married to Cem Karacal on January 24, 1984. Mother of two children, Sila Gizem born July 29, 1985 and Ada Irem born June 28, 1993.

Education: Graduated from Ankara Fen Lisesi, Ankara, Turkey, in June 1977; received Bachelor of Science Degree in Industrial Engineering from Middle East Technical University, Ankara, Turkey in June 1982; received Master of Science Degree in Computer Science from Oklahoma State University, Stillwater, Oklahoma in May 1988. Completed requirements for the Doctor of Philosophy degree at Oklahoma State University in May 1995.

Experience: Teaching Assistant, Industrial Engineering Department, Middle East Technical University, August 1982, to April 1984; Teaching Assistant, School ofIndustrial Engineering and Management, Oklahoma State University, January 1988, to December 1990; Member of Alpha Pi Mu, Tau Beta Pi and the Institute of Industrial Engineers.