ANALYSIS OF METHODS FOR DETERMINING WARM-UP LENGTH IN STEADY STATE SIMULATION

By

PRASAD SURESH MAHAJAN

Bachelor of Mechanical Engineering

PVG's College of Engineering and Technology

Pune, Maharashtra, India

2000

**Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May 2004**

ANALYSIS OF METHODS FOR DETERMINING

WARM-UP LENGTH IN STEADY STATE

SIMULATION

Thesis Approved:

[illegible]

Thesis Adviser

Michael Branson

Kenneth E Case

Alfred Carlozzi

Dean of The Graduate College

ACKNOWLEDGEMENTS

I wish to thank my major advisor, Dr. Ricki Ingalls for his precious and intelligent guidance. I would like to thank the other committee members, Dr. Kenneth Case and Dr. Michael Branson for their advice and cooperation. I would also like to express my deep gratitude towards Dr. Manjunath Kamath, Dr. Allen Schuermann, Dr. Jerry Banks, Dr. Lee Schruben, Dr. David Goldsman, Dr. Bruce Schmeiser, and Dr. Barry Nelson for their vital suggestions and advice.

I would also like to express special appreciation for my parents who supported me through out my studies and encouraged me to finish my thesis. Last, but not the least I would like to thank Dr. William Kolarik and the School of Industrial Engineering and Management for giving me this opportunity for pursuing the master's thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Simulation Modeling and Output Analysis

Simulation is a widely used tool by many organizations such as manufacturing and service to understand the behavior of a given system. It is also used to analyze the effects of applying different strategies to a system. Various strategies include applying different input distributions or changing system parameters like number of resources. It can either be used to analyze a current system by building a model and running it with actual input values (Example: arrival rate, number of servers), or it can be used to evaluate certain performance measures even before a system is built. Simulation can be used as a tool for finding bottlenecks, over-utilized resources, estimated waiting times and many more performance measures related to systems in the manufacturing and service industries.

Simulation models can be broadly categorized into two classes; deterministic and stochastic. Deterministic models use fixed, non-random values to specify the model and particular variant of the system. The output is fixed since randomness is not present. Stochastic models incorporate randomness as well as some elements of time elapsing [38]. This class of models is further classified into terminating and non-terminating (steady state) simulations. In the case of terminating simulations, the model runs for a finite horizon of time until some event occurs [8]. An example is the simulation of a bank which starts and stops at a fixed time. The output process is not expected to attain steady state behavior and the value of any parameter estimated from the output data will depend upon the initial conditions of the system. In the case of steady state simulations, the model runs for an extended amount of simulation time. The objective is to study the long-run behavior of the system. Generally, the objective of steady state simulation modeling is to obtain a steady state mean of the desired performance measure like, average number of parts per hour or average delay in queue, with a certain confidence interval having a pre-defined coverage probability.

A performance measure of a system is called a steady-state parameter if it is a characteristic of the equilibrium distribution of an output stochastic process [8]. An example is a continuously operating manufacturing system where the objective is to compute the average number of output produced per hour. This research will concentrate on the steady state simulation.

The primary purpose of most simulation studies is to obtain estimates of prescribed performance measures. Suppose that $Y_1, Y_2, Y_3, \ldots$ is the output process from a simulation run. Y is the steady state random variable of interest. Y may denote the average number in the system or delay in queue. If Y_i is the value of Y at i time unit(s), then

$$P\{Y \leq y\} = F_i(y) \text{ tends to } F(y) = P\{Y \leq y\} \text{ as } i \text{ tends to } \infty.$$

The purpose of steady state simulation is to estimate the mean of Y, denoted by μ, with a confidence level of $(1-\alpha)$ [24].

The process of simulation modeling involves a considerable amount of time and money. Important strategies and decisions are based on the results obtained. An obvious question one may ask is whether the output produced by simulation is reliable enough to make vital decisions? In other words, is the simulation output a true representative of what is going to be the actual?

1.2 Problems in Simulation Output

The following two problems may lead to erroneous results:

1. The input processes like the arrival rates and service times, driving the simulation are usually random variables. The resulting output is therefore random. Simulation results yield only estimates of measures of system performance. They are subject to sampling errors. Moreover the output $Y_1, Y_2, ..., Y_m$ in most of the cases

    - are not independent. The output is stochastic and autocorrelated. Positive autocorrelation results in overestimating the confidence interval where as negative autocorrelation results in underestimating the same
    - are not identically distributed
    - are not normally distributed

    The above-mentioned facts make it difficult to apply classical statistical techniques to the analysis of simulation output [10]. There is a vast amount of research done in this field, which gives practical methods to perform statistical analysis of output from discrete-event computer simulation.

2. It is generally not possible to choose the initial conditions for simulation to be representative of steady state. Most of the time, the system starts empty and idle. That is, the queues are empty and the servers are idle. These conditions do not necessarily represent steady state. Reconsider the output process $Y_1, Y_2, ..., Y_m$.

Any estimate, μ, based on observations $Y_1, Y_2, ..., Y_m$ obtained from simulation output will be biased [24]. This problem is called *the initialization bias problem* or *the start-up problem* in the simulation literature.

This thesis will concentrate on the initialization bias problem in steady state simulation. The following example illustrates the effect of initial transient on simulation output.

1.3 Example Model

In this example model, parts arrive according to an exponential distribution with a mean of 6 minutes. The arriving parts are divided into three types. 50% of the parts are type A, 30% are type B and remaining are type C. It is assumed that the initial condition of the system is empty and idle. The service times are exponentially distributed. There are 5 cells (machines), which process the parts in a sequence as shown in Figure 1.1. The measure of performance for this example is the average number of parts produced (throughput) per hour. The model is run for 5000 hours. Let Y_i (i = 1, 2, ..., m) represent the average number of parts produced per hour. Let m denote the run length in hours. The objective is to obtain an estimate of the steady state mean of the process Y_i. Let μ denote the actual steady state mean of this process. It is observed that the estimate of the mean of the process $Y_i = \overline{Y}_m \quad = \frac{1}{m}\sum_{i=1}^{m} Y_i \quad \neq \quad \mu$.

This is due to the effect of biased initial values. The graph in Figure 1.2 shows a plot of the performance measure "average parts produced" against "time". At the end of every hour, the value of average number of parts produced is recorded. It is observed that after an initial steepness the curve becomes steady and remains in that state until the end. The portion of the graph for which the curve appears to be approximately horizontal is said to be in steady state. The negative initialization bias can be easily observed from the graph.

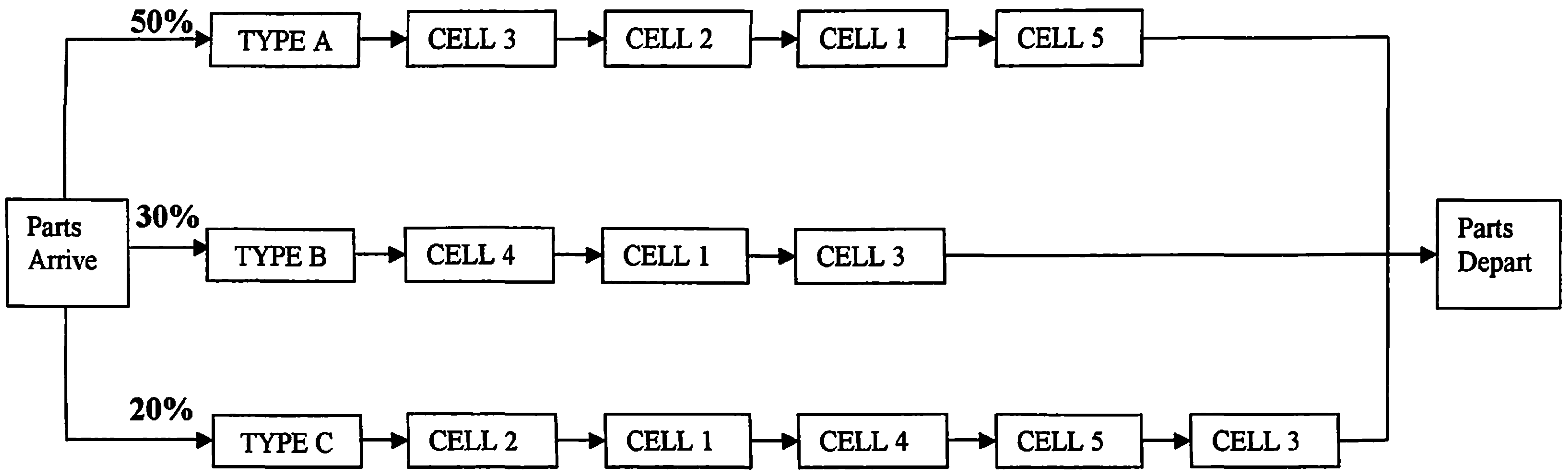


Figure 1.1: Job Shop Model

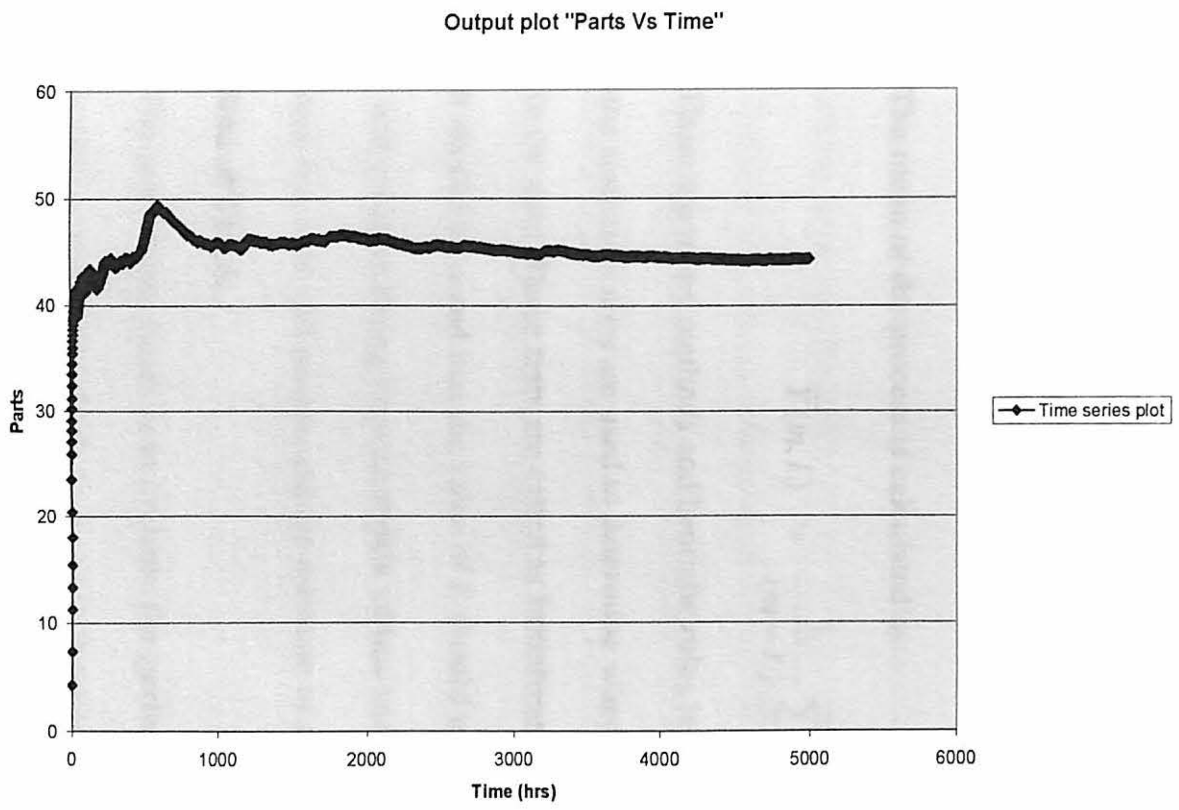


Figure 1.2: Graph of Average Parts Produced against Time in Hours

1.4 Overcoming the Initialization Bias Problem

There are two methods for removing/reducing initialization bias:

1. To run the simulation model with the initial conditions set to the steady state conditions. However, it is hard to predict the steady state conditions. Also, one needs to know the answer before running the simulation.
2. To run the simulation model for a period L called as warm up length, reset all statistics after time L and start recording observations for a period of m-L.

The mean of the process is calculated as:

$$\bar{Y}(m,L) = \frac{1}{(m-L)}\sum_{i=L+1}^{m} Y_i \approx \mu .$$

There are many methods and heuristic rules in literature for determining the value of L. Some statistical tests are used to determine whether or not initialization bias is present in the data. These tests are called as Initialization Bias Tests.

It should be noted that the value of L should not be too high or too low. If L is too large, it will result in losing important data values and a high variance of mean. If L is too small, some bias will still remain and the estimate of mean will be erroneous.

1.5 Outline of Thesis

The goal of this thesis is to evaluate the performance of some of the methods and heuristic rules in literature, which are used to determine the warm-up length L. The tests for initialization bias will not be evaluated. However, the literature review will involve the study of various methods, heuristic rules and tests for initialization bias.

This thesis will attempt to answer following questions: Which methods work under which conditions and which methods fail? If some methods work well for a particular condition, which one of them is the most effective? Are there any modifications, which when applied to the methods will improve their performance?

In order to evaluate their performance, the selected methods will be implemented on a simulation model for three different levels of utilization. The first level will have high traffic intensity, the second level will have moderate intensity and the third level will have low traffic intensity. Implementing a method on an experimental model for a given level of utilization will constitute one experiment. For each experiment the performance of the method will be evaluated based on some evaluation criterion. These criteria are discussed in Chapter 3.

The rest of the thesis is organized in four chapters. Chapter 2 will cover literature review. The detailed methodology, evaluation criteria, and experimental details will be explained in Chapter 3. Results will be documented in Chapter 4. Conclusion and suggestions will be presented in Chapter 5.

2. LITERATURE REVIEW

2.1 A Brief History of the Initialization Bias Problem

The problem of initialization bias has been addressed in the literature since the early days of simulation. There has always been a debate on whether to delete data or adjust the starting conditions in order to account for the bias. Early researchers had a mixed feeling on which approach to use. To assess the quality of performance measures (point estimators) obtained from simulation output, two characteristics were used, namely, "bias" and "variance". Bias measures systematic deviation from the true mean, whereas variance measures variation around the bias plus the mean [12]. Some authors have stated that data deletion is inappropriate because it results in an inflated variance of the estimator. Fishman (1972) [12] stated that truncation resulted in an inflated variance and a loss in statistical reliability for a model that he studied. According to Fishman (1972), bias and variance measure two separate characteristics of an estimator. So he suggested a single figure, the mean-square error (MSE) as the sum of variance and square of bias as a measure of point estimator quality. This is the most commonly used measure to judge the quality of point estimator. Snell and Schruben (1979) [36] and Kelton (1980) [20] showed that for a first order autoregressive process deletion increased or decreased the MSE depending on the simulation run length m and the warm-up length L. Blomqvist (1970) [5] has shown for $M/M/1$ queue (and certain other queuing systems) with m sufficiently large, that zero is that value of L which minimizes the mean squared error. If

$Y_1, Y_2, ..., Y_m$ is an output process for *M/M/*1 queue with utilization $\rho < 1$, the MSE of $\bar{Y}(m, L)$ is given by:

$$\text{MSE}[\bar{Y}(m,L)] = \text{E}\{[\bar{Y}(m,L) - \mu]^2\} = \{\text{Bias}[\bar{Y}(m,L)]\}^2 + \text{Var}[\bar{Y}(m,L)].$$

Law [24] considers the properties of a point estimator for some characteristic of the steady state random variable Y to evaluate the value of deletion. For example, while estimating $\mu = \text{E}[Y]$ from $Y_1, Y_2, ..., Y_m$, it is expected that

$$|\,\text{E}[(\bar{Y}(m,L) - \mu]\,| < |\,\text{E}[Y(m) - \mu]\,| \text{ for } L > 0\,.$$

Fishman (1972) showed that this is true for an autoregressive process of order one. It would also be true for the *M/M/*1 queuing process mentioned above. There are, however, situations where deletion could increase the bias. For an example, see the discussion of the simple inventory system and Figure 8.2 in Law and Kelton (1982) p.283 [26]. Deletion may also increase the variance of the point estimator. Fishman showed that for a first order autoregressive process, it does. However, if the observations at the beginning of a simulation run have particularly large variances, deletion could decrease the variance of the estimator [24].

Another approach to evaluate the effect of deletion is to assess the impact on the quality of confidence interval over the estimator. Deletion could result in degradation of the coverage of the confidence interval, if L is large relative to m. However, deletion has very small impact on the coverage of a confidence interval for *M/M/*1 queue according to results from Law (1975) [22], Law (1977) [23], Law and Kelton (1979) [25] and Law [24].

Law [24] suggests that deletion is generally advisable for replication and may give some improvement in point and interval estimator quality for methods based on a single long simulation run. The main problem here is how to estimate the value of L. Over the years, many authors have suggested various approaches to estimate L and to detect the initialization bias. Conway (1963) [9] suggested a heuristic rule to account for the bias by deleting a certain amount of data. Many heuristic rules were later developed which emphasized mainly on data deletion or re-starting the collection of statistics after a certain point in time. Each rule returned a value of time, say L, after which the expectation of the performance measures were very close to their limiting values. Such examples include the Conway Rule, Modified Conway Rule, Crossing-of-the-Mean Rule, Marginal Standard Error Rules (MSER) by White *et al.*, [41], MSER-5 by Spratt [37] and others. Gafarian, Ancker and Morisaku (1978) [15] evaluated a number of heuristic approaches. They concluded that some of the rules over estimate L whereas some of them underestimated the same. Some of them worked well for a very busy system, while they did not work well for the same system with a low utilization. Another study by Wilson and Pritsker (1978) [42] analyzed the effects of various combinations of initial condition rules and truncation rules. They concluded that a judicious selection of an initial condition was more effective than applying the truncation rules. Welch (1983) [39] suggested a graphical method based on ensemble averages. This method is very popular and easy to apply. It is a graphical method and the user determines the warm-up length by viewing a plotted graph. Kelton and Law (1983) [21] came up with a statistical method based on GLS regression. This method emphasizes data deletion and the determination of the warm-up length. Schruben (1982) [34] proposed a test for detecting initialization bias in the output. Goldsman, Schruben and Swain [16] developed many other methods in the mid-nineties. In the 1990s, many methods like the Randomization Test (Yucesan [44]), and new heuristics were proposed. Some of the methods included heuristic rule by White (1997) [40] based on the Marginal Confidence Rule (MCR). White states that the objective of truncation method should be to remove observations that are rare and atypical of individual observation sequences of a given fixed length. A comparison of five heuristic rules was carried out by White, Cobb and Spratt [37] in 2000. They concluded that the MSER-5 method works best for mitigating the bias where as the batch means method by Goldsman, Schruben and Swain [16] was least effective.

To date, research has not provided a single method which can accurately determine the warm-up period. This emphasizes the difficulty of the initialization bias problem. This difficulty arises because there is no definite way of defining steady state. Conway [9] states that equilibrium is a limiting condition that may be approached but actually never attained. For some systems, steady state may be approached at a geometric rate, where as for others, even after a generous run, the system may not be close to equilibrium. This makes the problem even more difficult to tackle.

The methods used to determine warm-up length can be classified into the following three groups:

1. Graphical
2. Statistical
3. Heuristic

Tests for initialization bias are used to indicate whether or not bias is present in the data. These methods can be used to detect the warm-up length with the help of a deletion strategy. This thesis is restricted to the above three groups of methods. However, a detailed study of some of the initialization bias tests is carried out and algorithms for implementing the tests are presented. These methods are presented in Table 2.2.

Statistical methods can be subdivided into two sub-groups, namely, basic statistical and advanced statistical tests. The advanced statistical tests involve time series analysis.

This research focuses on developing well-defined codes for six methods from the three groups and analyzing their performance. Advanced statistical tests will not be considered. The methods are listed in Table 2.1. A brief description of these methods is given in Section 2.2

Graphical	Welch's Method [39]
	Statistical Process Control Method (SPC) [32]
	Modified Statistical Process Control Method [33]
	Banks Carson and Nelson method [2]
	Method of Cumulative Sums [29]
Statistical	Randomization Tests [44]
	Method based of Generalized Least Squares Regression [20]
	Sequential Method [31]
	Scale Invariant Truncation Point Method [19]
	Simulation Run Control Method [18]
Heuristic	Conway Rule [9]
	Crossing of the Mean Rules [13]
	Marginal Standard Error Rule (MSER) [40], [37]

Table 2.1: List of Methods for Determining Warm-Up Period

Initialization Bias Test	Initialization Bias Test by Schruben [34], [35]
	Goldsman, Schruben and Swain (GSS) family of Tests [16]
	Other version of Schruben's Tests [29]

Table 2.2: List of Initialization Bias Tests

2.2 Review of Methods for Detecting Warm Up Length

The methods are classified into three types:

1. Graphical
    a) Welch's Method
    b) SPC Method (Robinson)
    c) Modified SPC Method (Robinson)
    d) BCN (Banks, Carson and Nelson) Method
    e) Cumulative Sum Method (Schruben)
2. Statistical
    a) Randomization Tests (Yucesan)
    b) Generalized Least Squares Regression Method (Kelton and Law)
    c) Sequential Method (Pawlikowski)
    d) Scale Invariant Truncation Point Method (Jackway and DeSilva)
    e) Simulation Run Control Method (Heidelberger and Welch)
3. Heuristic
    a) Marginal Confidence Rule (White)
    b) Conway Rule (Conway)
    c) Crossing the Means Rule (Fishman)

2.2.1 Graphical Methods

These methods include Welch's Method, Statistical Process Control Method (SPC), Modified SPC Method, Banks, Carson and Nelson Method of Batch Means deletion and Schruben's Cumulative Sums Plot. These methods involve plotting a graph and deciding the warm up length. The point on the graph beyond which remaining data points appear to be evenly distributed around an approximately horizontal line is noted. The warm up length is the simulation time corresponding to this point.

2.2.1.1 Welch's Method

This is the simplest and most general technique used for determining the warm-up length. It is a graphical technique that requires multiple replications. The Welch's method is explained in the following steps:

1. Make n replications of the simulation each with run length m. Let $Y_{i,j}$ be the i^{th} observation from the j^{th} replication. Thus i takes values from 1 to m and j from 1 to n.

2. Calculate the ensemble averages over the replications. These will be $\overline{Y}_i$'s where

$$\overline{Y}_i = \sum_{j=1}^{n} \frac{Y_{ij}}{n} \text{ for } i = 1, 2, \ldots, m.$$

3. Define a moving average $\overline{Y}_{iw}$ to smooth out the high frequency oscillations in $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m$. w is the window and is a positive integer. w is less than or equal to $m/4$. $\overline{Y}_{iw}$ is as follows:

$$\overline{Y}_{iw} = \frac{\sum_{s=-w}^{w} \overline{Y}_{i+s}}{2w+1} \quad \text{, for } i = w+1, w+2, \ldots, m-w$$

$$\overline{Y}_{iw} = \frac{\sum_{s=-(i-1)}^{i-1} \overline{Y}_{i+s}}{2i-1} \quad \text{for} \quad i = 1, 2, 3, \ldots, w.$$

4. Plot $\overline{Y}_{iw}$, for $i = 1,2, \ldots, m-w$ and choose L to be that value of i beyond which $\overline{Y}_{iw}$ appears to be converged. See Welch (1983) p. 292 [39] for an aid in determining

convergence. A FORTRAN code for this method is presented in Section 1 of Appendix A. Law and Kelton 2000 [27] suggest that $n \geq 5$ and $w \leq m/4$.

2.2.1.2 The Statistical Process Control (SPC) Method [32]

This approach uses the concepts from statistical process control to determine whether steady state has been reached or not and thus determine the warm-up length. In the case of SPC, the process is either "in-control" or "out-of-control". Applying the same analogy to simulation, it can be said that the process is either in transient state or steady state. When the process is in transience, it can be considered "out-of-control". Robinson [32] presents the following stages that describe the SPC Method.

Stage 1: Perform Replications and Collect Output Data

The method needs the simulation to be run for n replications. The observation interval can be one hour. The main assumption of SPC is that data are independent and normally distributed. Approximately normal and independent data are obtained by replicating the simulation run several times and taking means across the replications. The means across these replications, which are called as the ensemble averages are assumed to be normally distributed by the central limit theorem. Generally 5 - 10 replications are desirable for better results. The output is in the form:

$$\overline{Y}_i = \frac{\sum_{j=1}^{n} \overline{Y}_{i,j}}{n} \text{ for } i = 1, 2, \ldots, m$$

where n is the number of replications performed. m is the total number of observations made in each replication.

Stage 2: Test that the output Data Meet the Assumptions of SPC

As stated earlier, SPC is based on two assumptions. The data must be normally distributed and uncorrelated. The solution for this is to batch the ensemble averages $\overline{Y_i}$ into b batches of size k. To determine the batch size k, start with k =1. The size is increased until the autocorrelation goes below 0.1. The data should also be tested for normality. The K-S test given in Charles Ebeling [7] can be used to test the data for normality. This method is given in Section 1 of Appendix B. The batches are formed as follows:

$$y(k) = \left[\frac{\sum_{i=1}^{k} \overline{Y_i}}{k}, \frac{\sum_{i=k+1}^{2k} \overline{Y_i}}{k}, \ldots, \frac{\sum_{i=(b-1)k+1}^{bk} \overline{Y_i}}{k} \right].$$

The batch means are represented as: $y\ (k) = (y\ (1), y\ (2), \ldots, y\ (b))$

Stage 3: Construct a Control Chart for the Batch Means Data

Robinson (2002) recommends that the run length be at least four times the estimated length of the initial transient so the mean and standard deviation are calculated on steady-state data. The population mean and standard deviation are to be estimated from the last half of the data. The standard deviation is estimated as follows:

$$\hat{\sigma} = \sqrt{\frac{1}{(\frac{b}{2})} \sum_{l=b-[b/2]+1}^{b} s_l^2}$$

where s_l^2 is the standard deviation about the individual means in $y(k)$ and is calculated from the individual observations obtained from each replication.

There are two-control limits

1) Warning limits: $WL = \hat{\mu} \pm 1.96\hat{\sigma} / \sqrt{n}$

2) Action limits: $AL = \hat{\mu} \pm 3.09\hat{\sigma}/\sqrt{n}$

Stage 4: Identify the Initial Transient

In this stage, the initial transient is determined. The initial transient occurs until the control chart is out-of-control. Bissell (1994) [4] identifies rules to determine whether or not the process is out of control. Following are the rules [32]

- The time-series data violates an action limit
- Two consecutive values violate either the upper or the lower warning limit.
- Frequent values in relatively close succession that violate a warning limit.
- Persistent trend in the time-series data.
- A run of seven or more values on either side of the mean $\hat{\mu}$.
- Excessive zigzagging, with few points near to the mean and many near to the control limits.
- The warm-up period for the model can be selected by identifying the point at which the time-series data is in-control and remains in-control.

2.2.1.3 The Modified SPC Method (Robinson [33])

This is a new method by Robinson [33]. This is a slight modification of previous SPC method [32]. Like the previous method this method requires all four stages. However, the batch size is selected by carrying out a test for autocorrelation using the Von Neumann's test statistic and the test for normality is done using the Anderson-Darling Test. The method for testing for insignificant autocorrelation is similar to the method stated in Fishman 1996 [14]. Refer to Section 2 of Appendix B for a complete description of the Anderson-Darling used for this test. The Von Neumann's test for autocorrelation is described in Section 3 of Appendix B.

Another change in this method is in the number of control limits. Three sets of control limits are calculated

$$CL = \hat{\mu} \pm z\hat{\sigma} / \sqrt{n} \quad \text{for} \quad z = 1, 2, 3$$

Using these limits a control chart is constructed showing the mean and three sets of control limits.

The following rules are used to identify the transient [33]:

1. A point plots outside a 3-sigma control limit.
2. Two out of three consecutive points plot outside a 2-sigma control limit
3. Four out of five consecutive points plot outside a 1-sigma control limit
4. Eight consecutive points plot on one side of the mean
5. Initial points all plot to one side of the mean

The warm up length is that value of time at which the process is in control and remains in control. The process is considered in control when none of the conditions above exist.

2.2.1.4 BCN (Banks, Carson and Nelson Method of Cumulative Mean Deletion) [2]

This is a graphical method. It is based on plotting a graph of cumulative batch means and deleting batches till the process appears to be in steady state.

The procedure is described below:

1. Run the simulation for a length of m and make n replications.
2. Batch the data for each replication into b batches per replication of length k.
3. For each batch in a replication, calculate the batch mean

$$\overline{Y}_{i,j} \text{ for } i = 1, 2, \ldots, b \quad j = 1, 2, \ldots, n$$

Calculate the mean $\overline{Y}_i = \frac{\sum_{j=1}^{n} Y_{ij}}{n}$ for each i = 1, ..., b.

4. Calculate the cumulative means. Plot the cumulative mean data over time and observe the graph.
5. If initialization bias is present, then delete the first batch mean and then go to step 4. Otherwise there is no bias, go to step 6.
6. The warm-up length is the number of batch means deleted times the batch size.

2.2.1.5 Schruben's Cumulative Sum Plot

The algorithm for this method is given in *The Handbook of Industrial Engineers* [29].

Schruben suggested a plot called cumsum plot. This plot is sensitive to bias in the output and needs the simulation run to be one long replication.

Define $S_0 = 0$

$$S_j = \sum_{t=1}^{j} (\overline{Y} - Y_t) \text{ for } j = 1, 2, \ldots, m.$$

For no initialization bias, E[S_j] = 0. In this case, the plot will tend to cross zero at several time periods. But if initialization bias is present then the plot will be on one side of zero for a long time. Batching can be used to smooth the plot.

The algorithm given below appeared in Nelson [29]

1. Initialization: Length of simulation m; batch size k; number of batches b. $a \leftarrow 0$; Array $s[j] \leftarrow 0$ for $j \leftarrow 0, 1, \ldots, b$.
2. Data: $y[t]$, the t^{th} observation from a single replication.
3. Batching: Divide the data into b batches of size k.

4. Compute overall averages:

```
DO $j \leftarrow 1$ to $b$;
    $a \leftarrow a + y[j]$
End do
$a \leftarrow a/b$
```

5. Generate cumsum plot

```
DO $j \leftarrow 1$ to $b$:
    $s[j] \leftarrow s[j-1] + a - y[j]$
End do
```

6. Output: Plot $s[j]$ versus j.

If additional data is obtained new batch averages are calculated and following modifications are made

```
$s[j] \leftarrow s[j] + j*(y[b+1] - a)/(b+1)$ for $j \leftarrow 1, 2, \dots, b$
$s[b+1] \leftarrow 0$
$a \leftarrow (b*a + y[b+1])/(b+1)$
```

2.2.2 Statistical Methods

In statistical methods, the null hypothesis is that no initialization bias is present. A value of test statistic is calculated and compared to a critical value to decide whether or not to accept the null hypothesis.

2.2.2.1 Randomization Test

Yucesan (1993) [44] presented a method to detect the initialization bias. It is based on randomization tests. He formulated the problem of initialization bias in a hypothesis testing framework concerning the mean of the process. Randomization test is applied to test the null hypothesis that mean of the process is unchanged throughout the run. However the article does not consider the higher order effects of initialization bias such as change in process variance. The advantage of using this method is that, no assumptions, like that of normality are required. The steps needed to perform this test are summarized below:

1. Run the simulation for a length of time m hours.
2. Obtain an output time series $Y_1, Y_2, ..., Y_m$.
3. Batch the data into b batches of length k.
4. Obtain b batch means $\overline{Y}_1, \overline{Y}_2, \overline{Y}_3, ..., \overline{Y}_b$.
5. Partition the batch means into two groups. For the first iteration the first group must include the first batch mean and the second group should contain remaining b-1 batch means.
6. For each iteration, the grand means of the two groups are compared. If the difference between the two grand means is significantly* different from zero, the null hypothesis of no initialization bias is rejected.

   **To access the significance a distribution of difference is required. Since it is not known, randomization is used as explained in the next paragraph.*
7. By using randomization, an empirical distribution is obtained and the original observed difference is seen far in the tail.
8. If the hypothesis is rejected, the groups are rearranged; second batch is added to the first group and the second group will contain (b-2) batch means and step 6 is repeated.

9. If hypothesis is accepted then the group 2 data is the steady state simulation output.

Randomization Test

Eric V Noreen [30] states that

> Randomization is used to test the generic null hypothesis that one variable (or a group of variables) is unrelated to another variable (or group of variables). Significance is assessed by shuffling one variable (or a set) with respect to another variable (or a set). If the variables are related, then the value of the test statistic for the original unshuffled data should be unusual relative to the values of the test statistic that are obtained after shuffling.

Randomization can conveniently be used to test the null hypothesis that a given data has same distribution. The following steps are used:

- Place the first data point in group 1 and the remaining n-1 in group 2.
- Calculate the mean for group 1 and group 2 and find the absolute difference between the means d. d is the test statistic.
- Shuffle the data. The number of permutations will be $W= n! / n_1! * n_2!$, where n_1 is the number of points in group 1 and n_2 is the number of points in group 2. After each shuffle, calculate d. The number of times d is greater than or equal to the original values of d, is w. The probability value for a one-tailed test of the difference between the groups is given by $p = w/W$. If this value is $\leq$ the predefined significance value then the null hypothesis is rejected. The number of

permutations for a test can be less that W. This is called approximate randomization tests.

Randomization Test for Initialization Bias Detection

The algorithm for this test is given on the next page. The objective is to detect any significant change in the mean of the process.

The null hypothesis for this test is that there is no initialization bias in the output mean. Let the expected value of the process be $E[Y_i] = \eta_i = \eta(1-a_i)$ for i = 1, 2, ..., m. Schruben *et al.*, 1983 [35]. The function η_i is called the *transient mean function.* a_i*'s* reflect the arbitrary behavior in the mean of simulated process. When the process is asymptotically stationary, then $\lim_{i \to \alpha} a_i = 0$ and a_i's represent the changes in output due to initialization effects. $a_i = 0$ if no initialization bias is present. The null hypothesis H_0 is defined as:

$$H_0 : a_i = 0 \text{ for all } i.$$

Yucesan suggests batching the output data to reduce serial autocorrelation. Batching also smoothes high frequency fluctuations. It is difficult, however, to decide the size of the batch k. Law and Kelton [27] discuss difficulties in choosing k and b (Page 555). For this method, a batch size of k is used such that the autocorrelation among the batch means becomes < 0.5. Let the batch means be b_{m1}, b_{m2}, ..., b_{mb}. The number of shuffles needs to be decided before beginning randomization procedure. Let the number of shuffles be denoted by W. If the amount of data is very large, value of W will be large. However the test can also be conducted by predefining NS = 1000 or any other value less that W. This is called *approximate randomization tests.* Before starting the randomization, NS has to be set to some value. After NS is set groups 1 and 2 are set as discussed above. The test statistic d is calculated, which is absolute difference between means of group 1 and group 2. Every time the data is shuffled, the absolute difference between the means is calculated and is called the pseudo statistic d_s. Every time $d_s \geq d$, a counter *nge* is incremented by one. Initially *nge* is set to 0. After each shuffle, the value of *NS* is incremented by 1.

After *NS* reaches its predefined value, the ratio $(nge+1) / (NS + 1)$ is calculated, which is the *observed significance level* (*or the p value*) of the test. The null hypothesis is rejected if $p \leq$ specified rejection level for the test (p_s). If H_0 is rejected, the second batch mean is added to group 1 and new value of d is calculated. The procedure is repeated and p is calculated again. In this way the whole process is repeated until null hypothesis is not rejected. This is the truncation point.

The algorithm is as given below:

1. Batch the process into b batches of size k.
2. Calculate the batch means $\bar{Y}_1, \bar{Y}_2, \bar{Y}_3, \ldots, \bar{Y}_b$.
3. Check for autocorrelation with batches. If autocorrelation is > 0.5 increase batch size and go to step 2. Otherwise go to step 4.
4. Set the value of $NS = W$ and p_s to 0.05. Set $i = 0$.
5. Let $G_1 = \{ \bar{Y}_1 \}$ and $G_2 = \{ \bar{Y}_2, \bar{Y}_3, \ldots, \bar{Y}_b \}$.
6. Set k=1. If $k > b$ go to step 16.
7. Set the counter *nge* to 0, *shuffle counter* = 0.
8. Increment *shuffle counter* by 1.
9. Compute the actual test statistic $d = |\text{mean}(G_1) - \text{mean}(G_2)|$.
10. If *shuffle counter* > *NS* go to step 14. Otherwise go to step 11.
11. Shuffle the data and calculate the pseudo statistic d_s.

12. If $d_s \geq d$ then $nge = nge+1$.

13. *shuffle counter* $=$ *shuffle counter* $+1$. Go to step 10.

14. Compute the significance level $p= (nge+1/NS+1)$.

15. $k=k+1$.

    If $p \leq p_s$ reject the null hypothesis and G_1: $G_1 \cup \{\bar{Y}_k\}$ $G_2 = G_2 \setminus \{\bar{Y}_{k-1}\}$ go to step7.

    If $p \geq p_s$ accept the null hypothesis, set $i=1$ and go to step 16.

16. If $i = 0$, then the bias is still present, increase the data size and restart the test.

    If $i = 1$, then the initialization bias is over and its length is k-1 batch means

The code for this method is given in Section 2 of Appendix A.

2.2.2.2 Kelton and Law GLS Method

Kelton, (1980) [20] proposed a regression based procedure to reduce or remove bias owing to artificial starting conditions. This method is also explained in Kelton and Law (1983) [21]. Method of deletion can be used to remove the initialization bias [20]. It can markedly improve statistical validity with only minor loss in efficiency if any criteria (in particular the confidence interval coverage probability) other than mean square error are considered. Kelton (1980) [20], gave a detailed examination of the effect of several tactical alternatives (including deletion) on several statistical performance criteria, when a process is generated by a non-stationary first order autoregressive model. The results convey that the technique of deletion can be an effective and efficient way of dealing with the initialization bias problem. As per Kelton (1980) [20], the true coverage probabilities of nominal 90% confidence intervals can often be brought close to the desired level by initial deletion without unduly widening them. Moreover, the quality of point estimator is only slightly impaired and in some cases it is enhanced. Despite the fact that Kelton showed these results for AR(1) only, he found those results encouraging enough to use deletion.

Consider *M/M/*1 queuing process. Let Y_i, $i = 1, 2, \ldots, m$ be the i^{th} delay in queue. Let μ be the steady state mean of this process. If $E[Y_i]$ is plotted against i then this function will approach μ as i is increased. The function $E[Y_i]$ against i, is called as *transient expectation function* (TEF) of process Y_i. The goal, as stated by Law and Kelton, for removing the bias is to obtain averages of points which have expectations near μ. If n replications of the process Y_i are made and averages across replications (ensemble averages) are computed, then $\overline{Y}_i$ are defined as:

$$\overline{Y}_i = \frac{\sum_{j=1}^{n} Y_{ij}}{n}.$$

$\overline{Y}_i$'s can be used as a proxy TEF. If $E(Y_i) \cong \mu$ for $i \geq L$ then in econometric parlance [21]

$$\overline{Y}_i = \mu + \eta_i$$

where η_i's are random variables with $E[\eta_i] = 0$.

If a line is fitted through $\overline{Y}_i$'s then such a line should be flat for $i > L$. Thus if a line is fitted, depending on the flatness it can be said whether $\overline{Y}_i$ is close to μ or not. It is thus a test for the flatness of TEF. A difficulty with this idea of fitting a line is that $\overline{Y}_i$'s correlated which is contrary to the assumption of classical regression. Thus ordinary least squares regression cannot be used to test the data for zero slope. Instead, Kelton suggests using the generalized least squares (GLS) regression. Kelton and Law use the method by Amemiya [1].

It has been shown by Kelton and Law using a model and an experiment that GLS is definitely better and more essential than OLS. The idea is to start at the end of $\overline{Y}_i$ series for the initial fit, and then move the segment backwards towards the beginning of the data until it appears that the TEF is no longer flat, as evidenced by the rejection of the null hypothesis of zero slope [21]. If the line fitted initially to the end segment of the data has a slope significantly different than zero then the length m of the simulation run has to be increased. Kelton and Law also suggest batching of data into b batches of length k and calculating b batch means. These batch means then form the points on which regression is performed.

The following notations are used for describing the procedure as given in Kelton and Law [21]:

n= the number of replications.

m_0= the initial length of each of the n replication.

Δm= the number of points added to each of the n replications, if necessary.

m^* = the maximum replication length.

b= the number of batches.

p^*= the maximum initial deletion proportion.

p_o= the minimum initial deletion proportion.

β= the size of the test for zero slope.

f = the maximum number of segments over which a fit is made, including the initial fit.

The procedure is as follows [20]:

The initial line is fitted to the last 100*(1-p^*) % of the batch means data. If the initial zero slope test does not result in rejection, the interval is moved backwards towards the beginning of the time series. This is done by reducing the deleted proportion by an amount $\Delta p = (p^*-p_0) / (f-1)$. So, the next line is fitted through $(p^*-\Delta p)b$ to $(1-\Delta p)b$ batch means. In other words the right end point is not kept fixed, rather it is moving along with the left end point. If the zero slope test for this segment fails then the next line is fitted through points $(p^*-2\Delta p)b$ to $(1-2\Delta p)b$. As long as the zero slope test fails, the deletion proportion is diminished by Δp until rejection occurs or the deletion proportion reaches p_0. At most f fits will be done and the interval moves back by constant amount $(\Delta p)b$. It is important that m be exactly divisible by b.

The following steps are given in Kelton and Law [21]

Step 1: Make n independent replications, each of length m_0 points.

Step 2: Average over the replication to obtain a time series $Y_1, Y_2, \ldots, Y_{m_0}$. Let $m=m_0$

Step 3: Group the m points into b batches of length k each. Compute b batch means.

Step 4: Fit a straight line through batch means $p^*b + 1, \ldots,$ b (by Amemiya's GLS procedure), and perform a test for zero slope at level β.

a) If the test fails to reject the null hypothesis of zero slope, go to step 5.

b) If the step indicates rejection, then:

i. If $m+\Delta m \leq m^*$, then $m \leftarrow m+\Delta m$, and go to step 3. In this case, each of the n replications must be continued for an additional Δm points.

ii. If $m+\Delta m > m^*$, display a message that m^* is too small, set $p=p^*$, and go to step 7.

Step 5: Let $\Delta p = (p^*-p_0)/(f-1)$, and let $p= p^*-\Delta p$.

Step 6: Fit a straight line through batch means $pb+1, \ldots, (p+1-p^*)b$, and perform a test for zero slope at level β.

a) If the test fails to reject:

1. If $p-\Delta p \geq p_0$, then $p \leftarrow p-\Delta p$, and go to step 6.
2. If $p-\Delta p < p_0$, then go to step 7.

b) If the test indicates rejection, then $p \leftarrow p+\Delta p$, and go to step 7.

Step 7: Let $L = pm$ (to the nearest integer), and return with L and m.

Note: The manner in which the replication length is extended is by arithmetic rather than geometric. Step 6 indicates that the initial as well as the final points are moved back. This makes the length of the segment constant.

2.2.2.3 Sequential Procedure for Steady State Simulation

Pawlikowski (1990) [31] presented sequential tests based on method of spectral analysis and non-overlapping batch means to stop the simulation experiment when the required relative precision of confidence intervals is achieved. The procedure is preceded by testing for initialization bias. Thus, the experiment consists of two stages:

Stage 1: Detect length of initial transient period.

Stage 2: Steady state behavior is simulated and analyzed.

Pawlikowski [31] presents a sequential procedure based on a stationarity test proposed by Schruben, *et al.*, (1983) [35]. It is used to test the hypothesis that a sufficient number of initial transient data has been (or has not been) discarded. This test is carried out in a fashion similar to any classical statistical test. The value of a chosen test statistic is calculated and compared to a standard statistic. The hypothesis that a sufficient data has been discarded is rejected or accepted at an assumed significance level α. The procedure is as follows:

1. Apply any one of the heuristic rules as described in Pawlikowski [31] to get an initial approximation for the truncation point n_0^*.
2. The steady state estimator for variance $\sigma^2[\overline{Y}(n)]$ and the number of degrees of freedom for its chi-square distribution κ is obtained using the last n_v observations from the remaining n_t values. N_v is usually ≥ 200. Obtain: $\hat{\sigma}^2[\overline{Y}(n)]$ and κ. This can be obtained using various methods presented in Fishman (1973 p.289) [13], Fishman (1971) [11], Heidelberger and Welch [17] and Schruben (1982) [34].
3. Test the first n_t observations for stationarity. The procedure is given in Pawlikowski [31].
4. If the test accepts the hypothesis of no initialization bias then the procedure is stopped.
5. If the test rejects the hypothesis more observations are discarded from the beginning and the same amount of observations are added at the end.

   Steps 2-5 are repeated iteratively.

2.2.2.4 Time Scale Invariance Method

Paul Jackway and Basil DeSilva [19] present a methodology for detecting the steady state of a discrete-time stochastic process. This method is optimized for an exponential transient. According to the authors, For many queuing systems, the rate at which a queue converges to its steady state characteristics, independently of the system's initial state, eventually becomes (for large values of time t) dominated by an exponential term of the form $\exp(-t/\tau_Y)$ where τ_Y is a characteristic of the queuing system.

τ_Y is called as the relaxation time. It is a characteristic time constant in the equation $x = \mu - (\mu - x_0)\exp(-t/\tau_y)$. This equation is a convergence form of a first order differential equation, which describes a system approaching a steady state μ with a rate proportional to its present distance from that steady state. According to the authors the value 4τ is the time at which the system is close (within 2%) to its steady state value.

A bias function is obtained as shown below. By scaling in time by τ_y and considering the simulation output as a realization of a process with a standardized exponentially decaying bias function

$$E[X_i] = \mu - B(i/\tau_y) = \mu - B(t_i)$$

$$t_i = i / \tau_y.$$

The bias function is given by

$$B(t) = \mu \exp(-t).$$

The quantity τ_y is seen as time scale parameter, which relates sampling rate to the relaxation rate [19].

The data is partitioned into batches of length $b^* = 4\tau_y$. This way the initial transient is fit inside the batch (Yi: $i=0, 1, 2, \ldots, b^*-1$).

The bias in batch j is given by

$$B_j(t_i) = e^{-4j}\mu\exp(-t_i) \text{ for } i = 0, 1, 2, \ldots, b^* - 1 \quad \text{and } j = 0, 1, 2, \ldots$$

The batch size is increased from a very low value to the point where the above-mentioned conditions are satisfied. This is the truncation point.

Making modifications to Schruben's test, 1983 Schruben *et al.*, [35], the authors derive a test statistic and its variance estimate.

$$T \approx \sum_{k=1}^{b} k\, S_k \exp\left(\frac{-4k}{b}\right)$$

S_k is the cumulative sum process.

$S_k = \overline{Y}_n - \overline{Y}_k$, where $\overline{Y}_k = \frac{1}{k}\sum_{i=1}^{k} Y_i$.

Var $(T) \approx b^3\sigma^2/247$.

Under the null hypothesis $T' = 15.7T\,(b^3\,\sigma^2)^{1/2}$ has a standard normal distribution. Another test statistic can be formed if estimate of σ^2 with associated degrees of freedom v are known. This statistic is:

$$\hat{T}' = 15.7T(b^3\hat{\sigma}^2)^{-1/2}.$$

It has a student's t distribution with v degrees of freedom. The value of v and the estimate $\hat{\sigma}^2$ can be found out using Fishman 1973 page 289 [13]. The hypothesis of no initialization bias is rejected if $\hat{T}' > t(v,\alpha)$.

2.2.2.5 Simulation Run Control Method

Heidelberger and Welch [18] develop a method of run length control based on the sequential comparison of the relative half-width of a confidence interval at a pre-defined accuracy.

They present following parameters for the method:

1. j_{max} - maximum run length.
2. j_l - an initial checkpoint.

3. I - Multiplicative checkpoint increment parameter.

4. Relative half width requirement ε.

A stationary portion testing procedure is applied to $\{Y_{(n)}, (n = 1, 2, \ldots, j_1)\}$. This is described in detail in Heidelberger and Welch [18]. It is used to find n_0 (if any) such that the sequence $\{Y_{(n)}, (n = n_0+1, \ldots, j_1)\}$ is from a covariance stationary process.

If the sequence fails this testing procedure then the next checkpoint is considered and the test for stationary portion detection is carried out again. The next checkpoint is j_2 = minimum $\{I^*j_1, j_{max}\}$. If the sequence passes the test, a confidence interval is generated. This method is also described in detail in Heidelberger and Welch [18]. If the test passes and confidence interval is generated then the estimated half width of the confidence interval ERWH = confidence interval width / $2\overline{Y}$ where

$$\overline{Y} = \frac{1}{(j_1 - n_0)} \sum_{n=n_0+1}^{j_1} Y_n .$$

ERWH is then compared with ε. If ERWH $\leq \varepsilon$, the test is over and the simulation will stop; otherwise the simulation will proceed to next checkpoint and the procedure will be repeated again. For details on this test, refer to Heidelberger and Welch [18].

2.2.3 Heuristic Methods to Detect Initialization Bias

Many authors have developed rules of thumb to detect initialization bias (See Gafarian, *et al.*, [15], K. Preston White [40], Conway [9] and Pawlikowski [31]). Some rules like Conway and modified Conway are very conservative [15]. Gafarian, *et al.*, [15] studied five rules with respect to mean values on *M/M/*1 queuing system and found that none of them worked satisfactorily and should not be recommended for practitioners. Wilson and Pritsker have carried out a similar study. Few rules are explained here, some of which will be tested.

2.2.3.1 Marginal Standard Error Rules (MSER and MSER-5)

The MSER by White (1997) [40] and MSER-5 by Spratt 1998 [37] determine the warm-up length that optimally balances the tradeoff between loss of precision due to reduction in sample size and gain in accuracy due to elimination of bias. These rules select the value of d that minimizes the marginal confidence interval. White [40] presents this marginal confidence rule (MCR) for resolving the initialization bias problem. In his paper, White compares four different rules by applying them to output sequences generated by ten runs each of four representative queuing simulations. Results show that simple heuristics can reduce initialization bias. Many previous rules were based on mean square error to select a truncation point. White proposes to select a truncation point based on minimum width of the confidence interval about the truncated sample mean. He gives the following rationale:

> If the observations later in the output sequence as a whole provide a good estimate of the central tendency of the output at steady state, then initial observations should be truncated to the extent that these diminish our confidence in that estimate. Thus we will seek to mitigate bias by removing initial observations that are far from the sample mean, but only to the extent this distance is sufficient to compensate for the resulting reduction in the calculation of the confidence interval half width.

For a finite stochastic process $\{Y_i(j)\colon i=1, 2, ..., n\}$ the optimal truncation point is given by

$$d(j)^* = \underset{n>d(j)\geq 0}{\arg\min}\left[\frac{z_{\alpha/2}\, s(d(j))}{\sqrt{n(j)-d(j)}}\right]$$

where $z_{\alpha/2}$ is the value of the unit normal distribution associated with a $100(1 - \alpha)$ percent confidence.

For a fixed confidence level, $z_{\alpha/2}$ is a constant. The expression then, can be written as

$$d_j^* = \underset{n > d_j \geq 0}{\arg\min} \left[\frac{1}{(n(j) - d(j))^2} \sum_{i=d+1}^{n} (Y_{i(j)} - \overline{Y}(n,d)_{(j)})^2 \right].$$

For a given output sequence $d(j)^*$ is determined by solving the unconstrained minimization problem defined by the above equation. The MSER heuristic is applied to the raw data where as the MSER-*m* rule is applied to *b* batch means where *b* = run length/batch size (*m*).

2.2.3.2 Conway Rule [9]

Conway 1963 suggests the following rule to truncate the initial data in order to reduce bias. "Truncate a series of measurements until the first of the series is neither the maximum nor the minimum of the remaining set."

This is done for a few pilot runs to decide upon a stabilization period. After this is done, the period is deleted from the result of each run.

Following algorithm is constructed using the steps given in Gafarian *et al.*, [15].

1. Decide *n* and *m* the number of exploratory replications and the length of the exploratory replications.
2. Compute $y_{ik}^{+} = \max(y_{ik}, y_{i,k+1}, \ldots, y_{im})$ and $y_{ik}^{-} = \min(y_{ik}, y_{i,k+1}, \ldots, y_{im})$
3. For $k = 1, 2, \ldots, m$ determine t_i such that $y_{it_i}^{-} < y_{it_i} < y_{it_i}^{+}$ occurs for the first time.
4. Estimate of the truncation point t^* is given by $\max\{t_1, t_2, t_3, \ldots, t_n\}$

The code for this method is given in Section 3 of Appendix A.

2.2.3.3 Crossing the Means Rule

This rule is stated in Fishman (1973) p. 285 [13]. This rules states that

> Compute the running cumulative mean as data are generated. Count the number of crossings of the mean, looking backwards to the beginning. If the number of crossings reaches a pre-specified value, which means you have reached the truncation point.

Following algorithm is based on steps given in [15]:

1. Generate the simulation output $\{Y_1, Y_2, \ldots, Y_m\}$
2. Define $w_j = \begin{cases} 1, \text{ if } Y_j > \overline{Y}_m,\ Y_{j+1} < \overline{Y}_m \text{ or } Y_j < \overline{Y}_m,\ Y_{j+1} > \overline{Y}_m \\ 0, \text{ otherwise} \end{cases}$

   $j = 1, 2, \ldots, m\text{-}1$

   $\overline{Y}_m = \frac{1}{m}\sum_{j=1}^{m} Y_j$.
3. The number of times the series crosses the mean is given by $\Omega_m = \sum_{j=1}^{m-1} w_j$.
4. Calculate $\Omega_1, \Omega_2, \ldots, \Omega_d$ such that at d the number of crossings is equal to the pre-specified number.

   The code for this method is given in Section 4 of Appendix A.

Other Rules

Many other rules are given in Wilson and Pritsker [43], Gafarian, *et al.*, [15] and Pawlikowski [31].

2.3 Review of Initialization Bias Tests

Initialization bias tests are statistical tests used to determine whether the data contains bias or not. The null hypothesis is that there is no initialization bias. The test is performed and a test statistic is calculated. This test statistic is compared to the critical value and the null hypotheses is either accepted or rejected. In this thesis the following tests for initialization bias are reviewed:

a) Initialization bias test by Schruben as it appears in *The Handbook of Industrial Engineers* [29]

b) Goldsman, Schruben and Swain (GSS) family of tests

c) Other version of Schruben's tests

These methods are used to detect the presence of initialization bias. They do not tell how much data to delete. A deletion strategy that specifies the amount of data to delete is required in order to use these tests for determining the warm-up length.

2.3.1 Initialization Bias Tests by Schruben

Schruben (1982) [34] developed a test based on the concept of standardizing the stochastic simulation output process to represent "noise" in which "a signal" due to initialization bias may be detected. According to Schruben the simulation output may be conceptualized as a continuous time stochastic process, $\{Z_t;\ 0 < t < \infty\}$. Z_t represents the "state" of the simulation at time t.

The stochastic simulation output can be considered in the form

$$Y_i = \mu_i + X_i; \qquad \text{for } i = 1, 2, \ldots, n$$

where t_i is the time at which Y_i is observed. The run is started at t_0.

As mentioned in Schruben (1982) [34], the stochastic process X_i is a real valued function of the process

$$\left\{ \int_{t_{i-1}}^{t_i} Z_s d_s ; i = 1, 2, \ldots \right\}$$

where μ_i is the unknown deterministic function. It represents a potential shift in the mean of the output process. E[X_i] = 0 and thus μ_i= E[Y_i].

Schruben assumes that $\{X_i\}$ is stationary and ϕ- mixing with finite variance. The property of ϕ- mixing is defined as follows in Schruben 1982 [34]:

> Let E be an event (with a positive probability of occurring) that depends only on the behavior of the process up to time t_k. Let F be an event that depends only on the behavior of the process after time t_{k+n}. That is, n time units pass between possible occurrences of event E and event F. The process is ϕ- mixing if the supremum (over k, E and F) of |Prob(F|E) – Prob(F)| is bounded by a real valued function of n, ϕ_n, with $\lim_{n\to\infty} \phi_n = 0$. Intuitively, the distant future behavior of the process (event F) is almost independent of the present or past behavior of the process (i.e. event E).

Refer to Chapters 20 and 21 of Billingsley 1968 [3] for an explanation on the theory for dependent stochastic processes. No assumptions about the function μ_i are made. The procedure involves standardizing the output series and analyzing it as a "signal" due to a changing mean function μ_i in the presence of "noise" due to X_i. A limiting stochastic process called as a standard Brownian bridge process is used for standardization. $\mathscr{B}_t$ is a process similar to the standard normal random variable. Brownian bridge process has continuous sample paths. It has the following four properties:

1. $\mathscr{B}_c = \mathscr{B}_t = 0$,
2. $E[\mathscr{B}_t] = 0, 0 \leq t \leq 1$,
3. $\text{Cov}(\mathscr{B}_{t}, \mathscr{B}_{t2}) = \min(t_1, t_2)\,(1\text{-}\max(t_1, t_2))$, and,
4. Sets of $\mathscr{B}_t$ have a jointly normal distribution.

Schruben presents a sequence of partial sums on which the tests can be based. The sequence is given by

$$S_n(k) = \overline{Y}_n - \overline{Y}_k \quad k = 1, 2, \ldots, n$$

$$S_n(0) = 0.$$

The sequence $S_n(k)$ contains the differences between the average of the entire output series $\overline{Y}_n$, and the average of the first k observations $\overline{Y}_k$.

Substituting the equation "$Y_i = \mu_i + X_i$" in the equation "$S_n(k) = \overline{Y}_n - \overline{Y}_k$", $S_n(k)$ can be expressed as a combination of two components. The first is a deterministic unknown signal and the other is a stochastic "noise" component.

Consider the equation

$$S_n(k) = M_n(k) + X_n(k).$$

The signal component is $M_n(k) = \overline{\mu}_n - \overline{\mu}_k$ and the noise component is

$$X_n(k) = n^{-1}\sum_{i=1}^{n} X_i - k^{-1}\sum_{i=1}^{k} X_i.$$

The noise process is scaled and its convergence to $\mathscr{B}_t$ is shown.

B_t is defined such that:

$B_t = tnX_n\,(tn)/(\sqrt{n}\sigma)$; t=1/n, 2/n, ..., 1.

It can be argued that $\lim_{n\to\infty} B_t = \mathcal{B}_t$.

B_t has the same properties as $\mathcal{B}_t$. The presence of signal $M_n(k) = \bar{\mu}_n - \bar{\mu}_k$ must be detected. As per Schruben (1982) [34]

> Scaling the magnitude of $M_n(k)$ and the run duration to the unit interval in the same manner as done to standardize the noise process results is a standard deterministic signal D_t given by
>
> $$D_t = tnM_n(tn) / \sqrt{n}\sigma; \quad \text{for } t = 1/n, 2/n, \ldots, 1.$$
>
> If μ_i is constant for the whole run, D_t becomes zero for all values of t.

Initialization bias tests can be based on the peak and location of signals. The signals are not zero. The peaks are usually at the beginning of the run.

As quoted in Schruben (1982): "the initialization bias test involves the analysis of the standardized test sequence $T_n(t) = [tn]S_n([tn]) / \sqrt{n}\sigma; t \in (0,1)$ with $T_n(0)=0$"

Let $\hat{t}$ denote the observed location of the (first) maximum in $\{T_n(t)\}$and $\hat{s} = \sigma T_n(t)$. The observed value of $\hat{x} = \hat{s}^2 / \sigma^2(\hat{t}(1-\hat{t}))$ is analyzed. If this value is unusual then the output series contained no initialization bias. Large values of $\hat{x}$ are regarded as unusual.

Test Procedure

As stated above, if no negative initialization bias is present, the large, positive maximum value of the scaled test sequence $T_n(n)$ is unusual. The probability is denoted by $\hat{\alpha}$. A large value of $\hat{\alpha}$ indicates no significant negative initialization bias. $\hat{\alpha}$ is the observed level of significance for this test. The null hypothesis is given by:

H_0 = the output process has a constant mean.

The hypothesis is rejected if $\hat{\alpha}$ is less than the specified probability, α, of rejecting a true hypothesis (α is the type I error level)

Define $\hat{h}$ such that

$$\hat{h} = \hat{s}^2 / (3\hat{\sigma}\hat{t}(1-\hat{t}))$$

where $\hat{h}$ will approximately have an F distribution with 3 and ν degrees of freedom. The following steps are found in Schruben (1982) [34]:

1. Find $\hat{s}$, the global maximum of $\{kS_n(k)/\sqrt{n}$ for k = 1, 2, ..., $n\}$ and its location $\hat{k}$. In case of ties use the first value.

2. Estimate $\hat{\sigma}^2$ and v using the following equations

   $$\hat{\sigma}^2 = \hat{\sigma}_\varepsilon^2(1 - \hat{\phi}_1 - \hat{\phi}_2 -, \ldots, \hat{\phi}_p)^{-2}$$

   $$\nu = (n\hat{\sigma}_\varepsilon) / (2\hat{\sigma}(p - \sum_{i=1}^{p}(p-2i)\,\hat{\phi}_i).$$

   The latter portion of the simulation output is fit to a p-order autoregressive model and the above constants and error terms are estimated. (Refer to program in Appendix B of Fishman [13]).

3. Set $\hat{t} = \hat{k}/n$ and compute $\hat{h}$ using the above equation.

4. Compute $\hat{\alpha} = \overline{F}_{3,\nu}(\hat{h})$ where $\overline{F}_{3,\nu}(.)$ is the upper tail of the distribution function for an F variate with 3 and ν degrees of freedom.

5. Reject the hypothesis of no negative bias if $\hat{\alpha} < \alpha$.

2.3.2 Goldsman, Schruben and Swain (GSS) Initialization Bias Tests

Goldsman, Schruben and Swain [16] propose a family of tests to detect the initialization bias. These tests are a generalization and extension of earlier tests proposed by Schruben (1982) [34] and Schruben, Singh and Tierney (1983) [35].

Let $Y_1, Y_2, Y_3, \ldots, Y_m$ be the output of the simulation. It is assumed that this process has a transient mean function $\mu_i \equiv \mathrm{E}[Y_i] = \mu(1-a_i)$ for $i = 1, 2, \ldots, n$ where a_i's are constants. If bias is not present, then $\mu_i = \mu$. For the tests described here, divide the above process into two contiguous, non-overlapping portions. For each half, calculate the estimate of variance of sample mean. A large difference between these two estimates is unlikely if Y_i's are stationary. The null hypothesis is rejected if the difference in the estimates is significant. [16].

Cash, *et al.*, [6], explain the family of tests suggested by Goldsman, Schruben and Swain. The test statistic is a F statistic that compares the variability in the first portion of the output process to that in the other portion. The null hypothesis is rejected if $F > F_{1-\alpha,\, c,\, d}$ the $(1-\alpha)$ quantile of the F distribution with c and d degrees of freedom.

Batch Means Test [6]

The algorithm for this test is as given below:

- Divide the process into b non-overlapping batches of length k each.
- Define
  - $\overline{Y}_i = \frac{1}{k}\sum_{j=1}^{k} Y_{(i-1)k+j}$.
  - $Q_{BM} = k\sum_{i=1}^{b}\left[\overline{Y}_i - \frac{1}{b}\sum_{j=1}^{b}\overline{Y}_j\right]^2$.

- $V_{BM} = \dfrac{Q_{BM}}{b-1}$.

- Divide the total number of batches into two groups. Let V_{BM1} be the variance estimator of the first group and V_{BM2} be the variance estimator of the second group.
- Compute the ratio $F_{BM} = V_{BM1}/ V_{BM2}$.
- $F_{critical}$ for this test is $F_{1-\alpha,b`-1,b-b`-1}$.

Area Test [6]

The algorithm for the test is in Cash, *et al.*, [6]:

- Transform the data into b standardized time series and compute a variance estimator based on the area under the standardized time series as given below.
- $\overline{Y}_{i,j} = \dfrac{1}{j}\sum_{t=1}^{j} Y_{(i-1)k+t}$ for $i = 1, 2, ..., b$; $j = 1, 2, ..., k$.
- $T_{i,k}(t) = \dfrac{[kt](\overline{Y}_{i,k} - \overline{Y}_{i,[kt]})}{\sigma\sqrt{k}}$ for $i = 1, 2, ..., b$.
- $\hat{A}_i = \dfrac{\sigma}{k}\sum_{j=1}^{k}\sqrt{12}\,T_{i,k}(j/k)$ for $i = 1, 2, ..., b$.
- $Q_{AREA} = \sum_{i=1}^{b} \hat{A}_i^{\,2}$.
- $V_{AREA} = \dfrac{Q_{AREA}}{b}$.
- $F_{critical}$ for this test is $F_{1-\alpha,b`,b-b`}$.

The test statistic is given by:

$$F_A = V_{AREA1} / V_{AREA2}.$$

Maximum Test [6]

This test is based on the basis of location and magnitude of the maximum deviation. The test is presented for the presence of negative bias.

- $S_{i,j} = \overline{Y}_{i,k} - \overline{Y}_{i,j}$
- $\hat{K}_i = \arg\max_{1 \le j \le k} \{ jS_{i,j} \}$
- $\hat{S}_i = \hat{K}_i S_{i,\hat{K}_i}$
- $Q_{max} = \sum_{i=1}^{b} \frac{k\hat{S}_i^2}{\hat{K}_i(k - \hat{K}_i)}$
- $V_{MAX} = Q_{MAX} / 3b$
- $F_{critical}$ for this test is $F_{1-\alpha, 3b', 3b-3b'}$

  $F_{MAX} = V_{MAX1} / V_{MAX2}$

Combined Tests [6]

Two more F-tests can be created by combining the statistics of the batch means, area and maximum tests.

Area + Batch Means Test

$Q_{BM+AREA} = Q_{BM} + Q_{AREA}$

$V_{BM+AREA} = (V_{BM} + V_{AREA}) / 2b\text{-}1.$

The test statistic for this test is

$$F_{BM+AREA} = V_{(BM+AREA)1} / V_{(BM+AREA)2}.$$

The critical test statistic is

$$F_{2b'-1, 2b-2b'-1}.$$

Max + Batch means Test

$Q_{BM+MAX} = Q_{BM} + Q_{MAX}$

$V_{BM+MAX} = (V_{BM} + V_{MAX}) / (4b\text{-}1)$.

The test statistic for this is

$$F_{BM+MAX} = V_{(BM+MAX)1} / V_{(BM+MAX)2}.$$

The critical test statistic is $F_{4b'-1,4b-4b'-1}$.

2.3.3 Initialization Bias Test by Schruben

Nelson [29] explains another version of initialization bias test by Schruben. This test is to be applied after some amount of data has been deleted initially by applying any of the heuristic rules mentioned in the next section. In this test the simulation is run for a long time. The output obtained is split into two halves. The cumulative sum values from each half are compared in terms of the location and magnitude of their maximum deviation from zero. The null hypothesis is rejected if the behavior of the data in the first half is significantly different from the second half. The algorithm as given in Nelson [29] is:

1. Initialization: length of he replication m; batch size b; number of batches $k \leftarrow [m/b]$; arrays $a[j] \leftarrow 0$, $s[j] \leftarrow 0$, $smax[j] \leftarrow 0$ and $l[j] \leftarrow 0$ for $j=1, 2, \ldots$
2. Data: $y[t]$, the t^{th} observation from a single replication.
3. Batching: Divide the data into b batches of size k. $y[1], y[2], \ldots, y[b]$ will contain the batch means.
4. $n \leftarrow b/2$ The last batch is ignored if b is odd.
5. Calculate sample mean of each half

```
        $a_1=0$, $a_2=0$
        DO $i=1,n$:
         $a_1 = a_1 + y(i)$
```

```
    end do
    $a_1 = a_1/n$
    DO $j=n+1,2n$:
      $a_2 = a_2 + y(j)$
    end do
```

6. Locate the maximum for each half.

   If positive bias is suspected the n replace all >'s by <'s in this step.

```
    DO $i=1,n-1$
      $s[1] \leftarrow s[1]+a[1]-y[i]$
          If($s[1]>smax[1]$) then
              $l[1] \leftarrow i$ and $smax[1] \leftarrow s[1]$
          End if
    End do
```

7. Calculate the test statistic:

   (if $l[1] = 0$ test the bias for opposite sign. If $smax[2]$ is zero the m is too small.)

   $f \leftarrow l[2]*(n-l[2])*smax[1]/(l[1]*(n-l[1])*smax[2]*smax[2])$

8. If $f > F_{\alpha,3,3}$ then reject the hypothesis of no initial condition bias. (α is the significance level).

3. RESEARCH STATEMENT AND METHODOLOGY

As explained in the first chapter, the need for removing initialization bias is very important for obtaining true estimates of performance measures within the specified confidence interval. After a thorough review of literature, it can be said that data deletion is the fastest and the best method to remove initialization bias from the output as long as variance inflation is controlled and excess data is not deleted. Chapter 2 explains in detail a few of the many methods and heuristic rules suggested by various authors to detect bias or predict the warm-up length (truncation point). It has been observed that methods behave differently for different models. One method may work well for some models while it may not work so well for others. Also, it is seen that the same method gives different results for different performance measures. All of these issues contribute to the complexity of the initialization bias problem. There is a need to study the performance of these methods by varying the following:

1. System performance measures to be estimated.
2. Traffic intensity of the model.
3. Complexity of the model.

Such a study will form a guide for academicians as well as practitioners to conduct simulation runs using the method that best fits the model they will use. In the past, some authors have tried to design methods that decide not only the warm-up length but also the stopping point in a simulation run based on the quality of the point estimators

(Heidelberger and Welch [18]). Such an approach will be helpful only if the method used is the best one for the model as far as the complexity and the traffic intensity is concerned.

3.1 Research Objectives

The purpose of this thesis is to analyze the performance of six methods at varying levels of utilization. For this thesis the complexity of the model and the system performance measure to be estimated are held constant. The levels of utilizations are varied. The objectives of this thesis are:

1. To study and understand the working of different methods for determining the warm-up length in discrete event steady state simulation.
2. To prepare algorithms and/or codes for six methods so that they can be implemented.
3. To use these methods in determining the warm-up length for a simulation model.
4. Compare the performance of these methods based on following goodness criterion
   i. Mean square error (MSE)

      MSE = $(\theta - \hat{\theta})^2$ + variance ($\hat{\theta}$)

      where θ = Actual mean or the theoretical mean of simulation output data. Section 3.3.2 shows the procedure used to estimate the value of θ. $\hat{\theta}$ is the mean of the data left after deletion. If L is the warm up length and m is the run length, then

$$\hat{\theta} = \frac{\sum_{i=L+1}^{m} Y_i}{m - L}$$

$$\text{variance}\,(\hat{\theta}) = \frac{1}{(m-L)^2}\sum_{i=L+1}^{m}(Y_i-\theta)^2 .$$

m is the run length, L is the warm up length and Y_i, for $i = 1$ to m is the value of the performance measure at i time units.

A good method will yield a low value of MSE.

ii. Variance

By deleting data, the variance of the point estimator is might increase. Thus it is necessary to assess the quality of deleted data based on the variance of $\hat{\theta}$. This is calculated as

$$\text{variance}\,(\hat{\theta}) = \frac{1}{(m-L)^2}\sum_{i=L+1}^{m}(Y_i-\theta)^2 .$$

iii. Percentage change in mean square error

This calculates the percentage change in the initial mean square error (with no data deleted) after data deletion has been applied. The simulation output will have a MSE before data deletion has been applied. Let this be denoted by MSE_{ini}. After applying the warm up length and deleting L data points, MSE is calculated again. Let this be denoted by MSE_{fin}. Percentage change in MSE is given by

$$\%\ \text{change in MSE} = 100 * (MSE_{fin} - MSE_{ini}) \,/\, (MSE_{ini}).$$

A good method will always give a negative value of percentage change in MSE.

iv. Percentage change in variance

It is the percentage change in the initial variance after data deletion has been applied. Initial variance is the variance of the simulation output data with no warm-up. Let this be denoted by V_{ini}. Final variance is the variance of the data left after applying the warm-up length. This is calculated as given below

$$V_{fin} = \frac{1}{(m-L)^2} \sum_{i=L+1}^{m} (Y_i - \theta)^2$$

$$V_{ini} = \frac{1}{(m)^2} \sum_{i=1}^{m} (Y_i - \theta)^2 .$$

Percentage change in variance = 100* $(V_{fin}$ - $V_{ini})/ V_{ini}$.

v. Cost

Cost is calculated in terms of average computer time. This is the sum of computer time required to collect data and computer time required to perform the method. This is measured in seconds.

5. Draw conclusions based on observations of experimental results, revealing which method works well under the given traffic intensity and which does not.
6. Identify potential extensions to this thesis that will benefit simulation practitioners and academicians.

3.2 Scope of the Research

Most of the methods available in literature have been studied and understood. However this thesis will focus on the following six methods

1. Welch's Method
2. Modified SPC Method
3. Randomization Test
4. Conway Rule
5. Crossing the Means Rule

6. MSER-5 Rule.

Gafarian, *et al.*, (1978) [15] proposed the following measures of goodness to access the performance of a method for determining the warm-up length.

1. Accuracy: It is the ratio of value of warm-up length estimated by using a method ($E[L]$) to the theoretical value L_t, the point at which the system reaches steady state.
2. Precision: It is a measure of variation. It is given by the ratio

$$\frac{\sqrt{Var(\hat{L})}}{E[L]}.$$

3. Cost: Measured in terms of computer time required for using a particular method/rule. This consists of following:
    1. Computer time for the executing the code.
    2. Computer time for collecting output data only for a preliminary estimate of L, and subsequently discarding this data. [15]
4. Simplicity: It is the measure of easiness with which an average practitioner will be able to apply to a system simulation.

The measures 1, 2 and 4 above will not be used in this thesis because it is impossible to calculate E[L] and L_t. Also, since the measure of goodness "simplicity" is very subjective, it is not used. The four measures of goodness discussed in Section 3.1 will be used.

3.3 Experimental Setup

3.3.1 Experimental Models

The methods are tested on a simple job shop models. The basic model used in this thesis is shown in Figure 3.1. There are five cells, C_1, C_2, C_3, C_4 and C_5. Each cell has different number of machines (resources). There are three customer classes (three different types of parts). The overall arrival rate is Poisson (λ) hours and the service times for servers are exponential with means $\mu_{i,j}$ hours (i (*customer class*) = 1, 2, 3 and j (*cells*) = 1, 2, 3, 4, 5). The model is built and run in Arena 5 [45] simulation software.

The arriving parts are split into three types (*customer classes*), namely, type A, type B and type C with probabilities 0.5, 0.3 and 0.2 respectively. After splitting, their new arrival rates are the probability times the overall arrival rate. The parts get processed in the cells in the sequence shown in Figure 3.1. After being processed the parts exit the system.

The methods are applied for the same model with different utilizations. Three models with varying levels of traffic intensity are built by modifying the arrival rates.

Type I Model:

This model has a high level of utilization. The average utilization of all resources is close to 90%. The individual utilizations may vary from 80% - 95%.

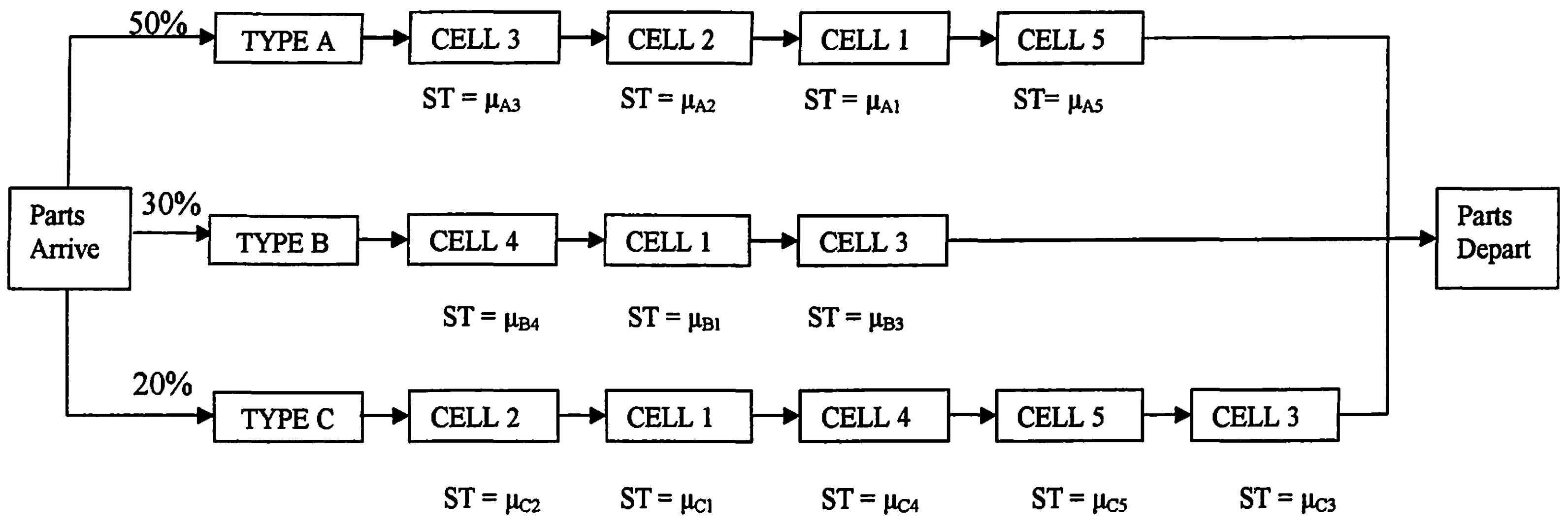
Type II Model:

This model has a moderate level of utilization. The average utilization of all resources is close to 70%. The individual utilizations may vary from 65% to 80%.

Type III Model:

This model has a low level of utilization. The average utilization of all resources is close to 50%. The individual utilizations may vary from 45% to 65%.

Tables 3.1, 3.2 and 3.3 give the data for these three models.



ST – service time

Figure 3.1: Model for the Experiment

Type I Model

Arrival Rate	Expo(4.5) minutes

Parts	Sequence	Processing Times (hours)
Type A	Cell C3	μ_{A3} = Expo(0.4)
	Cell C2	μ_{A2} = Expo(0.5)
	Cell C1	μ_{A1} = Expo(0.6)
	Cell C5	μ_{A5} = Expo(1.2)
Type B	Cell C4	μ_{B4} = Expo(1.2)
	Cell C1	μ_{B1} = Expo(0.8)
	Cell C3	μ_{B3} = Expo(1)
Type C	Cell C2	μ_{C2} = Expo(0.9)
	Cell C1	μ_{C1} = Expo(0.5)
	Cell C4	μ_{C4} = Expo(0.7)
	Cell C5	μ_{C5} = Expo(0.1)
	Cell C3	μ_{C3} = Expo(1.4)

Resources	Capacity
Cell C1	9
Cell C2	6
Cell C3	11
Cell C4	7
Cell C5	10

Average utilization = 92.56%

Table 3.1: Data for Type I Model

Type II Model

Arrival Rate	Expo(5.8) minutes

Parts	Sequence	Processing Times (hours)
Type A	Cell C3	μ_{A3} = Expo(0.4)
	Cell C2	μ_{A2} = Expo(0.5)
	Cell C1	μ_{A1} = Expo(0.6)
	Cell C5	μ_{A5} = Expo(1.2)
Type B	Cell C4	μ_{B4} = Expo(1.2)
	Cell C1	μ_{B1} = Expo(0.8)
	Cell C3	μ_{B3} = Expo(1)
Type C	Cell C2	μ_{C2} = Expo(0.9)
	Cell C1	μ_{C1} = Expo(0.5)
	Cell C4	μ_{C4} = Expo(0.7)
	Cell C5	μ_{C5} = Expo(0.1)
	Cell C3	μ_{C3} = Expo(1.4)

Resources	Capacity
Cell C1	9
Cell C2	6
Cell C3	11
Cell C4	7
Cell C5	10

Average utilization = 71.6%

Table 3.2: Data for Type II Model

Type III Model

Arrival Rate	Expo(8) minutes

Parts	Sequence	Processing Times (hours)
Type A	Cell C3	μ_{A3} = Expo(0.4)
	Cell C2	μ_{A2} = Expo(0.5)
	Cell C1	μ_{A1} = Expo(0.6)
	Cell C5	μ_{A5} = Expo(1.2)
Type B	Cell C4	μ_{B4} = Expo(1.2)
	Cell C1	μ_{B1} = Expo(0.8)
	Cell C3	μ_{B3} = Expo(1)
Type C	Cell C2	μ_{C2} = Expo(0.9)
	Cell C1	μ_{C1} = Expo(0.5)
	Cell C4	μ_{C4} = Expo(0.7)
	Cell C5	μ_{C5} = Expo(0.1)
	Cell C3	μ_{C3} = Expo(1.4)

Resources	Capacity
Cell C1	9
Cell C2	6
Cell C3	11
Cell C4	7
Cell C5	10

Average utilization = 51.68%

Table 3.3: Data for Type III Model

3.3.2 Estimating the actual value of performance measure θ:

The models are run for a very long time in the range of 10^6 - 10^9 hours. Values of the performance measure, say θ_i for these long runs are observed. The values of θ_i for each long run are within 1% of each other for all models. The average of θ_i's for all long runs is considered as a true estimate of θ for the given model.

For all models, the value of θ is also calculated theoretically using a queuing software *RAQS* [46].

For model type II and type III, the results obtained from *RAQS* [46] and from the averages of long runs are within 1%. For type I model, the results are within 3.5%. This verifies the results for all the models.

Values of θ for model types I, II and III calculated from the method described above are summarized in Table 3.4 below.

Model Type	Average Number in System (WIP)
Type I	119.41
Type II	35.16
Type III	22.71

Table 3.4: Actual Values of Performance Measures for all Models

3.3.3 Run Conditions

System is started empty and idle. Run length is 1000 hours. Considering that the model is a replication of a job shop, we propose to run the model for a quarter of a year. Assuming two eight-hour shifts per day and 5-day week, the run length is approximately 1000 hours.

3.3.3 Performance measure

Average hourly Number in system. Number in system (Inventory) is measured at the end of each hour and recorded. The average of all the recorded values is the average hourly number in system. This performance measure is chosen arbitrarily. Results may vary for different performance measures.

3.3.4 Levels of Utilization

Level 1: High traffic intensity (92.56%)

Level 2: Moderate traffic intensity (71.6%)

Level 3: Low traffic intensity (51.68%)

3.4 Methodology

The above objectives are converted into following stages:

Stage 1: Study various methods and rules mentioned in the literature for determining the warm-up length.

Stage 2: Thoroughly define the methods mentioned in Section 3.2 and wherever possible prepare programming codes to execute the algorithms.

Stage 3: Apply the methods and rules to each of the three simulation models. Estimate the value of warm-up length *L* for each method and rule.

Stage 4: Determine the measures of goodness mentioned in Section 3.1 for each method and rule at each of the three levels of utilization.

Stage 5: Draw conclusions based on the results of stage 4.

3.5 Procedure

The procedure for implementing the warm up length detection methods and calculating the goodness of measures is presented in form of a pseudo code.

The pseudo code to find measures of goodness for all six methods

```
Definition of variables
    $m$ → run length in hours
    $n$ → number of independent replications
    $L$ → warm-up length in hours
    $Y_i$ → Inventory in the system at time $i$; for $i = 1$ to $m$
    $\theta_a$ → Actual value of average hourly inventory
    $\theta_{ini}$ → Average hourly inventory calculated from simulation output before deleting
        any data
    $\hat{\theta}$ → Average hourly inventory calculated after deleting $L$ hours of data. This is an
        estimate of $\theta_a$
    initialB → Bias in $\theta_{ini}$ calculated against $\theta_a$
    finalB → Bias in $\hat{\theta}$ calculated against $\theta_a$
    initialVar → Average of variance of $\theta_{ini}$ for $n$ replications
    finalVar → Average of variance of $\hat{\theta}$ for $n$ replications
    initialMSE → Average of mean square errors on $\theta_{ini}$ calculated against the theoretical
                 value for $n$ replications
    finalMSE → MSE of $\hat{\theta}$ calculated against the theoretical value
    Max → represents the maximum run length required amongst all methods
Initialize variables
    iteration = 0
    n = 5
start
        IF (iteration = 0) THEN
                m = 1000
        end if
        DO i = 1 to 6
                Select method i
                        IF (the method requires single run) THEN
                                GOTO run
                        ELSE
                                GOTO multiple run
                        END IF
run:
                Run the model for m hours with n replications
                $finalMSE_{(i)} = 0$
                $initialMSE_{(i)} = 0$
                $finalVar_{(i)} = 0$
                $initialVar_{(i)} = 0$
                        DO j = 1 to n
                                Consider m points from $j^{th}$ replication
```

```
                  Calculate initial mean $\theta_{ini(j)} = \sum_{p=1}^{m} \overline{Y}_p$
                  Calculate initial variance $Var(\theta_{ini\,(j)}) = \frac{1}{m^2}\sum_{p=1}^{m}(Y_p - \theta_a)^2$
      apply method:
                  Apply method i to m data points
                      IF (method requires more data) THEN
                          $m = m + 200$
                          GOTO apply method
                      end if
                  $m\,(i, j) = m$
                  Find the warm up length
                  Warm up length = $L_{(j)}$ hours
                  Calculate mean and the variance of the remaining data
                  named as
                  $\hat{\theta}_{(j)} = \sum_{p=L_{(j)}+1}^{m} Y_p$
                  $Var\,(\hat{\theta}_{(j)}) = \frac{1}{(m - L_{(j)})^2}\sum_{p=L_{(j)}+1}^{m}(Y_p - \theta_a)^2$
                  Calculate the final bias $finalB_{(j)} = \hat{\theta}_{(j)} - \theta_a$
                  Calculate final MSE as
                  $finalMSE_{(j)} = finalB_{(j)}^{\ 2} + Var\,(\hat{\theta}_{(j)})$
                  Calculate the initial bias $initialB_{(j)} = \theta_{ini(j)} - \theta_a$
                  Calculate initial MSE as
                  $initialMSE_{(j)} = initialB_{(j)}^{\ 2} + Var(\theta_{ini(j)})$
                  $finalVar_{(i)} = finalVar_{(i)} + Var\,(\hat{\theta}_{(j)})$
                  $initialVar_{(i)} = initialVar_{(i)} + Var\,(\theta_{ini(j)})$
                  $finalMSE_{(i)} = finalMSE_{(i)} + finalMSE_{(j)}$
                  $initialMSE_{(i)} = initialMSE_{(i)} + initialMSE_{(j)}$
              END DO Loop j
          $initialMSE_{(i)} = MSE_{ini(i)} = initialMSE_{(i)}\ /\ n$
          $finalMSE_{(i)} = MSE_{fin(i)} = finalMSE_{(i)}\ /\ n$
          $initialVar_{(i)} = V_{ini(i)} = initialVar_{(i)}\ /\ n$
          $finalVar_{(i)} = V_{fin(i)} = finalVar_{(i)}\ /\ n$
          Percentage Change in MSE for method i =
              $100 * (MSE_{fin\,(i)} - MSE_{ini\,(i)})\ /\ (MSE_{ini\,(i)})$
          Percentage Change in Var for method i =
              $100 * (V_{fin\,(i)} - V_{ini\,(i)}) / V_{ini\,(i)}$

          GOTO finish
  multiple run:
          Run the model for n replications
          $finalMSE_{\,(i)} = 0$
```

```
                  $initialMSE_{(i)} = 0$
                  $finalVar_{(i)} = 0$
                  $initialVar_{(i)} = 0$

apply method II:  Apply the method i
                     IF (method requires more data) THEN
                           increase run length by 200
                           new run length $m_1 = m + 200$
                           GOTO apply method II
                     ELSE
                           Return warm up length L hours
                           $m_1 = m$
                     END IF
                           $m(i, j) = m_1$
                     DO j = 1 to n
                           Consider m points from $j^{th}$ replication
                           Calculate initial mean $\theta_{ini\,(j)} = \sum_{p=1}^{m} \overline{Y}_p$
                           Calculate initial variance $Var(\theta_{ini\,(j)}) = \frac{1}{m^2}\sum_{p=1}^{m}(Y_p - \theta_a)^2$
                           Calculate $\hat{\theta}_{(j)} = \sum_{p=L+1}^{m_1} Y_p$
                           Calculate $Var(\hat{\theta}_{(j)}) = \frac{1}{(m_1 - L)^2}\sum_{i=L+1}^{m_1}(Y_i - \theta_a)^2$
                           Calculate the final bias $finalB_{(j)} = \hat{\theta}_{(j)} - \theta_a$
                           Calculate final MSE as
                           $finalMSE_{(j)} = finalB_{(j)}{}^2 + Var(\hat{\theta}_{(j)})$
                           Calculate the initial bias $initialB_{(j)} = \theta_{ini(j)} - \theta_a$
                           Calculate initial MSE as
                               $initialMSE_{(j)} = initialB_{(j)}{}^2 + Var(\theta_{ini(j)})$
                           $finalVar_{(i)} = finalVar_{(i)} + Var(\hat{\theta}_{(j)})$
                           $initialVar_{(i)} = initialVar_{(i)} + Var(\theta_{ini(j)})$
                           $finalMSE_{(i)} = finalMSE_{(i)} + finalMSE_{(j)}$
                           $initialMS_{(i)} = initialMSE_{(i)} + initialMSE_{(j)}$

                     END DO Loop j
                  $initialMSE_{(i)} = MSE_{ini(i)} = initialMSE_{(i)} / n$
                  $finalMSE_{(i)} = MSE_{fin(i)} = finalMSE_{(i)} / n$
                  $initialVar_{(i)} = V_{ini(i)} = initialVar_{(i)} / n$
                  $finalVar_{(i)} = V_{fin(i)} = finalVar_{(i)} / n$

                  Percentage Change in MSE for method i =
                     $100 * (MSE_{fin\,(i)} - MSE_{ini\,(i)}) / (MSE_{ini\,(i)})$
```

```
                Percentage Change in Var for method i =
                    $100 * (V_{fin\,(i)} - V_{ini\,(i)}) / V_{ini\,(i)}$
                GOTO finish
finish:
        END DO Loop i

if (iteration = 0) then
    Max = 0
        DO p = 1 to i
                DO q = 1 to n
                        IF (m (p, q) > max) THEN max = m (p, q)
                END DO
        END DO
    m = max
end if
    iteration = iteration + 1
        If (iteration = 2) THEN
                GOTO end
        ELSE
                GOTO start
        END IF
end:
```

For this thesis, the value of *n* is chosen to be 5. It is advisable to have number of replications between 5 and 10. We chose to run the models for 5 replications. However, results may vary for different number of replications.

4. EXPERIMENTAL RESULTS

The methods mentioned in Section 3.2 can be broadly classified into two classes. The first class of methods requires multiple replications where as the second class only requires one run. The Modified SPC Method, Conway Rule and Welch's Method are classified as class one methods. Randomization Test, MSER-5 Rule, and the Crossing the Means Rule fall under class two methods. Run length for all experiments is fixed to 1000 hours initially. Some algorithms might require more data if the initial run length is not enough. In that case the run length is incremented by 200 hours. Within each of these two classes, some methods require the data to be batched. These are the Randomization Test, Modified SPC Method and the MSER-5 Rule. The MSER-5 Rule uses a fixed batch size of 5 observations. The Modified Statistical Process Control Method uses the Von Neumann's Test and Anderson-Darling Test to determine batch size. This method demands an increase in run length if the number of batches becomes less than 20 after either of the two tests has failed. Yucesan [44] requires increasing the batch size until the absolute serial autocorrelation falls below 0.5 for the Randomization Test.

After applying all of the methods on all of the models, the maximum run length amongst all the experiments is noted. This is denoted by m_{max}. All the models are then run for m_{max} hours and new output is obtained. All six methods are applied again to this data and new results are obtained.

4.1 Implementation Specifications

4.1.1 Welch's Method

Welch's Method requires three parameters to be specified. The run length m, the number of replications n and the window size w. The run length is fixed to 1000 initially. Law and Kelton [27] suggest taking the minimum value of w that so that graph appears smooth. For all the models value of w = 10 was enough to obtain a smooth graph. Welch's procedure is a graphical procedure. The user decides the warm-up length by observing a graph of averaged values against time. Thus there is subjectivity involved in this procedure. To minimize this subjectivity, twenty five users were asked to observe the graphical output and give their values for the warm-up length. Finally, the warm-up length was calculated as the average of 25 values.

4.1.2 Conway Rule

This rule also requires multiple replications. The number of replications is set to 5 and the run length is set to 1000 hours. This rule does not require any batching or tests for autocorrelation or normality.

4.1.3 Modified Statistical Process Control Method

This method requires the data to be batched with a batch size such that the data is approximately normally distributed and has negligible serial autocorrelation. Initial batch size is kept 1. The batch size is doubled if either the test for autocorrelation or normality fails. The number of batches is at least 20. If the number of batches falls below 20, then the test demands more data. For this thesis, if the test demands more data, the run length is incremented by 200 and the test is re-started. A graph of batch means is plotted with the control limits as defined in Section 2.2.1.3. The rules to determine whether the process is in control or not, are applied to the plotted data. The simulation time associated with the point beyond which the process is in control is the warm-up length.

4.1.4 Crossing the Means Rule

This rule requires the user to decide the value of number of crossings. Gafarian, *et al.*, (1978) [15] used a value of three. The same value is used here. The run length is set to 1000 hours. This rule does not require any batching or tests for autocorrelation or normality. The test is stopped when the number of times the process crosses the cumulative mean equals 3.

4.1. 5 Marginal Standard Error Rule-5

This rule requires a batch size of 5. Tests for autocorrelation or normality are not required.

4.1.6 Randomization Tests

This method requires the data to be batched. The initial batch size is taken to be one. The data is tested for autocorrelation. If the autocorrelation is > 0.5, then the batch size is doubled. The value of p is taken to be 0.05. If the test statistic d is > 0.05, then the system is in steady state. For each iteration, NS is set to $n! / n_1! * n_2!$, where

n= total number of batch means

n_1 = number of batch means in first group

n_2 = number of batch means in second group.

4.2 Summary of Results with Run Length = 1000 Hours

Results for all the three types of models are presented in Tables 4.1, 4.2 and 4.3. The initial values row is based on the models with no warm-up length. This is the base-line for calculating the percentage increase in variance and percentage increase in MSE.

4.2.1 Type I Model

For the system with high utilization the Modified SPC Method and the Randomization Test reduce the MSE and the variance. The Modified SPC Method gives the lowest MSE and the variance. For the methods that do not require multiple replications, the Randomization Test reduces the MSE by approximately 19% and the variance by 29%. It gives a comparatively higher MSE than the Modified SPC Method and a very low variance. These two methods require slightly more time to run than the other methods. However, both of these methods do not work with a run length of 1000 hours. They demand more data in order to give results.

Given that the run length is only 1000 hours, the Crossing the Means Rule and Welch's Method reduce the MSE by approximately 17% and 33% respectively. However they slightly increase the variance. Crossing the Means Rules takes slightly more time than the Welch's Method.

The Conway Rule and MSER-5 Rule take less time to perform, but increase the MSE. MSER-5 Rule increases the variance where as Conway Rule reduces the variance slightly.

MSER-5 Rule is computationally the most efficient.

4.2.2 Type II Model

This model is moderately utilized. In case of methods which do not require multiple replications, the MSER-5 Rule reduces both, the MSE and variance. It reduces the MSE by about 28% and variance by 3% approximately. The values for MSE and variance are very low. The MSER-5 Rule is the fastest to execute

The Crossing the Means Rule is also efficient but it increases the variance slightly. It reduces the MSE considerably (by 23% approximately). The randomization test increases both the MSE and the variance.

For methods that require multiple replications, the Conway Rule reduces the variance and increases the MSE. It requires less time to perform. The Conway Rule gives the minimum warm up length of 3 hours. The Modified SPC Method returns a warm up length of 25 hours. It also requires the longest time to execute. The Modified SPC Method increases both the MSE and the variance by approximately 3 and 2 percent. It gives a higher value of MSE compared to the initial value. The Welch's Method requires less time to perform; it reduces the MSE and increases the variance.

4. 2.3 Type III Model

This model is a low utilization system. There is negligible bias present in the simulation output data itself.

For methods that require multiple replications, the Modified SPC Method reduces the MSE and variance considerably. The Conway Rule reduces the variance but slightly increases the MSE. It requires less time to perform. The Welch's Method reduces the MSE but increases the variance. It takes less time to run

For methods that do not need multiple replications, the Randomization Test and MSER-5 Rule are effective in reducing MSE and the variance. However due to a large number of batch means, the Randomization Test requires extremely long time to perform. This method is computationally inefficient. On the other hand the MSER-5 Rule is most efficient. The Crossing the Means Rule increases both the MSE and variance.

Type I model (average utilization = 92%)								
Method	Average Warm up length (hours)	Average Run length (hours)	Final Mean	Final MSE	Final Variance	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Welch (1)	292	1000	129.0853	213.3846	1.8777	8	-33.42	13.33
SPC (1) (3)	0	5200	118.7974	72.3188	0.2984	127	-77.44	-81.99
Conway (1)	5	1000	129.7395	332.6243	1.6383	11	3.78	-1.12
MSER-5 (2)	221.8	1000	135.6506	395.7777	1.9550	5	23.49	17.99
Randomization (2) (3)	192	1480	127.0272	259.8652	1.1773	21	-18.92	-28.95
Crossing the means (2)	153.8	1000	128.7625	266.0326	1.7685	22	-16.99	6.74
Initial (before truncation)	0	1000	129.2340	320.4976	1.6569	5	0.00	0.00

Table 4.1: Results Summary for Type I Model with Run Length = 1000 Hours

(1) These methods need 5 replications, hence the warm-up lengths are same for all 5 runs

(2) These methods need 1 replication. They are applied 5 times to 5 runs. The values of warm-up length are average of 5 runs.

(3) The Modified SPC Method does not work for run length of 1000. It demands an increase in run length.

(4) The Randomization Test does work for run length of 1000. It demands an increase in run length.

Type II model (average utilization = 71%)								
Method	Average Warm up length (hours)	Average Run length (hours)	Final Mean	Final MSE	Final Variance	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Welch (1)	107	1000	35.5384	1.27	0.067	8.5	-34.67	5.51
SPC (1)	25	1000	35.6997	2.01	0.065	13.2	3.33	2.00
Conway (1)	3	1000	35.6606	2.02	0.063	12.1	3.70	-0.75
MSER-5 (2)	33	1000	35.4387	1.41	0.062	5	-27.80	-2.99
Randomization (2)	9.6	1000	35.6719	2.01	0.064	10.6	3.15	0.41
Crossing the means (2)	43.2	1000	35.5586	1.48	0.065	7	-23.78	1.93
Initial (before truncation)	0	1000	35.6096	1.95	0.064	5	0.00	0.00

Table 4.2: Results Summary for Type II Model with Run Length = 1000 Hours

(1) These methods need 5 replications, hence the warm-up lengths are same for all 5 runs

(2) These methods need 1 replication. They are applied 5 times to 5 runs. The values of warm-up length are average of 5 runs.

Type III model (average utilization = 51%)								
Method	Average Warm up length (hours)	Average Run length (hours)	Final Mean	Final MSE	Final Variance	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Welch (1)	74	1000	22.5866	0.070	0.021	11	-7.59	8.52
SPC (1)	432	1000	22.5859	0.062	0.026	31	-36.25	-4.88
Conway (1)	2	1000	22.6198	0.060	0.019	15	0.64	-7.78
MSER-5 (2)	6	1000	22.6182	0.061	0.019	9	-0.02	-5.64
Randomization (2)	4.4	1000	22.6201	0.060	0.019	750	-0.24	-7.21
Crossing the means (2)	25.8	1000	22.5971	0.069	0.020	25	2.18	6.04
Initial (before truncation)	0	1000	22.6002	0.065	0.019	8	0.00	0.00

Table 4.3: Results Summary for Type III Model with Run Length = 1000 Hours

(1) These methods need 5 replications, hence the warm-up lengths are same for all 5 runs

(2) These methods need 1 replication. They are applied 5 times to 5 runs. The values of warm-up length are average of 5 runs

Modified SPC Method, when applied to replication three, requires a run length of 5200 hours. All models are run for 5200 hours and new output data is obtained. The six methods are applied to the new data. The results are summarized in Section 4.3.

4.3 Summary of Results with Run Length = 5200 Hours

Results for all the three types of models are presented in Tables 4.4, 4.5 and 4.6. The initial values row is based on the models with no warm-up length. This is the base-line for calculating the percentage increase in variance and percentage increase in MSE.

4.3.1 Type I Models

For models that require multiple replications the Modified SPC Method gives a long warm up length and it reduces both the MSE and variance considerably. It requires a fair amount of time to run (33 seconds). The Conway Rule requires the longest time to execute. It gives almost the same MSE and a slightly lower variance than the corresponding values for the base model. The Welch's Method reduces both the MSE and the variance by a very small amount. It also takes slightly longer time to perform (33 seconds).

For the methods which do not need multiple replications, MSER-5 Rule reduces both the MSE and the variance. It is computationally most efficient. Randomization Test also successfully reduces the MSE but increases variance negligibly. The Crossing the Means Rule needs long time to perform. It reduces the MSE by approximately 7% and increases the variance negligibly.

4.3.2 Type II Models

Type II model is a moderately utilized system. For the methods that require multiple replications none of the methods reduce MSE. The Conway Rule reduces the variance by 0.17%. It also requires less time to perform. The Modified SPC Method gives a warm up length of 256 hours. It increases both the MSE and variance. The Welch's Method also increases the MSE and variance but it takes very less time to run.

Randomization Test requires the longest time to run. It increases the MSE slightly and gives almost the same variance. The Crossing the Means Rule reduces MSE by and variance by very small amount. MSER-5 Rule reduces the MSE and the variance. MSER-5 Rule is the most efficient to execute.

4.3.3 Type III Models

The Welch's Method and the Modified SPC Method increase both the MSE and the variance. For the Modified SPC Method the increase is considerably large. This method requires slightly more time to perform. However the Conway Rule reduces both MSE and variance and requires lesser time to run than most of the methods.

The Randomization Test reduces both MSE and variance. The Randomization Test requires significant amount of time to run. This is because for this low utilized system the autocorrelation is less and the algorithm produces more batch means resulting in an incredibly large number of iterations for calculating the test statistic. The MSER-5 Rule and the Crossing the Means Rule do not change MSE and variance much.

This model has very low initial MSE and variance. Randomization Test requires the longest time to perform. This is because of a large number of batch means leading to more number of iterations. The MSER-5 Rule and the Crossing the Means Rule require the minimum amount of time to perform (approximately < 20 seconds).

Type I model (average utilization = 92%)								
Method	Average warm up length (hours)	Average Run length (hours)	Final Mean	Final MSE	Final Variance	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Welch (1)	50	5200	119.2607	70.6416	0.2982	33	-2.32	-0.05
SPC (1)	768	5200	116.3985	53.7927	0.3349	33	-83.22	-79.79
Conway (1)	5	5200	118.8842	72.3384	0.2977	54	0.03	-0.24
MSER-5 (2)	195	5200	117.2861	51.6606	0.2907	16	-28.57	-2.59
Randomization (2)	256	5200	118.1012	44.0937	0.2997	31	-39.03	0.44
Crossing the means (2)	154	5200	118.3857	67.1267	0.2993	39	-7.18	0.31
Initial (before truncation)	0	5200	118.7974	72.3188	0.2984	15	0.00	0.00

Table 4.4: Results Summary for Type I Model with Run Length = 5200 Hours

(1) These methods need 5 replications, hence the warm-up lengths are same for all 5 runs

(2) These methods need 1 replication. They are applied 5 times to 5 runs. The values of warm-up length are average of 5 runs.

Type II model (average utilization = 71%)								
Method	Average Warm up length (hours)	Average Run length (hours)	Final Mean	Final MSE	Final Variance	Average Computing Time (Seconds)	% Change in MSE	%Change in variance
Welch (1)	65	5200	35.3248	0.0954	0.0112	18	4.83	0.26
SPC (1)	256	5200	35.3161	0.1115	0.0116	39	22.52	3.72
Conway (1)	3	5200	35.3485	0.0942	0.0111	21	3.48	-0.17
MSER-5 (2)	73	5200	35.2621	0.0798	0.0111	15	-12.35	-0.74
Randomization (2)	9.6	5200	35.3496	0.0957	0.0111	145	5.19	0.01
Crossing the means (2)	43.2	5200	35.3210	0.0906	0.0111	29	-0.43	-0.07
Initial (before truncation)	0	5200	35.3388	0.0910	0.0111	15	0.00	0.00

Table 4.5: Results Summary for Type II Model with Run Length = 5200 Hours

(1) These methods need 5 replications, hence the warm-up lengths are same for all 5 runs

(2) These methods need 1 replication. They are applied 5 times to 5 runs. The values of warm-up length are average of 5 runs.

Type III model (average utilization = 51%)								
Method	Average Warm up length (hours)	Average Run length (hours)	Final Mean	Final MSE	Final variance	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Welch (1)	66	5200	22.711	0.0175	0.00394	19	6.01	1.25
SPC (1)	3184	5200	22.688	0.0250	0.00979	32	51.45	151.47
Conway (1)	2	5200	22.713	0.0165	0.00389	16	-0.31	-0.13
MSER-5 (2)	6.25	5200	22.713	0.0167	0.00390	15	0.78	0.01
Randomization (2)	3	5200	22.714	0.0165	0.00389	1795	-0.05	-0.09
Crossing the means (2)	28.5	5200	22.709	0.0166	0.00391	27	0.60	0.48
Initial (before truncation)	0	5200	22.709	0.0165	0.00389	14	0.00	0.00

Table 4.6: Results Summary for Type III Model with Run Length = 5200 Hour

(1) These methods need 5 replications, hence the warm-up lengths are same for all 5 runs

(2) These methods need 1 replication. They are applied 5 times to 5 runs. The values of warm-up length are average of 5 runs

5. ANALYSIS OF RESULTS AND CONCLUSION

The evaluation procedure presented in this thesis is easy to implement on various methods and can be efficiently used to test the performance of the same. The measures of goodness used clearly indicate the quality of the methods/ heuristic rules.

In Section 5.1, an attempt is made to answer the questions that were posed in Section 1.5: Which methods work under which conditions and which methods fail? If some methods work well for a particular condition, which one of them is/are the most effective? Are there any modifications, which when applied to the methods will improve their performance? Initially, the methods are compared on the basis of model types used. In the latter paragraphs the overall performance of each method is presented based on when it fails and when it does not. The possible causes for failure of some of the methods are also documented. Finally modifications which may result in better performance of some of the methods are presented.

5.1 Discussion

A method is said to perform well if it reduces the MSE and variance and is computationally efficient. The results presented in Chapter 4 show that performance of methods varies with the type of model and the simulation run length (crossing the means rule being an exception for run length).

The Type I model is highly utilized. For a run length of 1000 hours, the MSE without data deletion is 320.49 and the variance is 1.6569. This shows that there is high initialization bias. None of the methods when applied to this model reduce both the variance and MSE with a run length of 1000 hours. The Modified SPC and the Randomization Test do not give any results for 1000-hour run length. They demand an increase in run length because the test for autocorrelation and/or normality fails. But, given more data, both of these methods reduce MSE and variance. The Modified SPC Method requires 5200 hours run length and the Randomization Test requires 1480 hours of average run length to perform. This makes the two methods computationally inefficient and also need availability of more data. The MSER-5 Rule performs poorly for Type I model with 1000 hours run length. However, this rule is highly efficient. It gives results in fraction of a second. Conway Rule is aggressive and performs poorly. It reduces variance but slightly increases the MSE without demanding more data. No method performs well for this type of model given that the run length is 1000 hours. If computational cost is not a concern and more data is readily available, the Modified SPC Method and Randomization Test are recommended.

When this model is run for 5200 hours, the MSE without data deletion is 72.31 and the variance is 0.298. Modified SPC Method and MSER-5 Rule performs the best for these conditions. None of the other methods perform well. MSER-5 Rule is more efficient than Modified SPC Method. For highly utilized systems, it is recommended to use the MSER-5 Rule and keep the run length long or use Modified SPC Method. Randomization Test reduces MSE by approximately 39% and does not change variance significantly. The Crossing the Means rule also reduces the MSE and does not significantly change the variance.

The Type II model is moderately utilized. The initial MSE and variance are very low; equal to 1.95 and 0.064 respectively. For a run length of 1000, only MSER-5 Rule works well. Conway Rule, Randomization Test, Modified SPC Method, Welch's Method and Crossing the Means Rule perform poorly for this model with a run length 1000 hours.

When the run length is increased to 5200 hours, the MSER-5 Rule and the Crossing the Means Rule perform well. All other methods perform poorly.

Type III model has a very low utilization. There is negligible initialization bias and low variability. For this type of a system a method that either reduces MSE and the variance or maintains the change in MSE and variance within 1% of the base model, is considered to perform well. For a run length of 1000 hours the MSER-5 Rule, Randomization Test and Modified SPC Method and the Conway Rule work well. For a run length of 5200 hours, the Conway Rule, Randomization Test and MSER-5 Rule and Crossing the Means Rule give good results. However, the Randomization Test is computationally very expensive. Modified SPC Method and Welch's Method perform very poorly.

To conclude the above discussion it can be said that the Welch's Method does not work well for most of the models The Welch's Method is highly subjective, hence different results may be observed for different group of users.

The Crossing the Means Rule appears to work well for low and moderately utilized systems with longer run lengths. However it gives the same warm up length as that for a run length of 1000 hours for both the models. The improved results are due to better values obtained from simulation data and not because of better performance of the method.

The MSER-5 Rule works well for longer run lengths, regardless of the model used. For smaller run lengths, it works well for models with moderate and low utilization.

The Conway Rule seems to work well for low utilized models where the initialization bias is negligible. This rule is very aggressive. It gives the minimum warm-up length for all models and run lengths.

The Modified SPC Method performs poorly for moderate utilization systems. For the moderate and low utilized models with long run length, the variance in the data is very low. So the control limits on the process are very close. Because of this, more points lie outside the 1-sigma limit. For a run length of 5200 it gives a very high warm up length on the Type III model. The data violates the rules on two occasions. For the Type III model, on the first occasion, the warm up length is 160 hours and on the second it is 3184 hours. This is because at the end of the batch means plot 1 additional point falls outside the 2σ limit. Taking the first warm-up length gives better results. If some rule can be added to neglect such outliers or re-start the test taking more data when the warm up length exceeds 25% of the run length, then the performance of this method can be even better. For the high utilized model, it does not work well with a long run length; but for smaller run lengths there is a high possibility that this method will demand more data due to high variability and autocorrelation. Given more data it is the most effective for such models.

The Randomization Test works well for Type I and Type III models under certain conditions. For high utilization systems, if the run length is small there is a considerable possibility that this method will demand for more data. This is because due to the presence of autocorrelation in the data. Given more data, it works well for Type I models.

Also, for Type III models which generate data with negligible autocorrelation and having long run lengths, the Randomization Test takes an incredibly long time to run. The following modifications are suggested to reduce the run time for low utilized systems:

- Minimum number of batches should be 20.
- Approximate randomization tests are recommended for low utilization models where the expected run time is very high.

  Results achieved using above modifications are presented in Table C.1 of Appendix C. It can be observed that the computational time is reduced drastically and better results on MSE and variance are obtained.

Tables 5.1, 5.2 and 5.3 list the methods/ rules applicable for the three model types used in this research.

Highly utilized system	
Name of the method	**Recommendation**
Modified SPC Method	Use long run length
MSER-5 Rule	Use long run length
Randomization Test	Start with small run length and use only if additional data can be generated

Table 5.1: Recommended Methods for Highly Utilized Systems

Moderately utilized system	
Name of the method	**Recommendation**
MSER-5 Rule	-

Table 5.2: Recommended Methods for Moderately Utilized Systems

Low utilized system	
Name of the method	**Recommendation**
Randomization Test	Use the modifications presented in Appendix C
Conway Rule	-
MSER-5 Rule	Use long run length

Table 5.3: Recommended Methods for Low Utilized Systems

5.2 Future Research

The testing of methods needs to be implemented on different range of models using different performance measures. However, it is hard to compute MSE by the method used in this research since the theoretical mean of the point estimator is generally not known. The current research focused on performance of the methods on systems with varying utilization levels. A more thorough study needs to be done, varying the complexity of the model and changing the performance measures.

Tests for initialization bias were not analyzed and compared because no deletion strategies were suggested by the authors. Performance of these methods needs to be assessed by developing various deletion strategies. Methods using time series are not tested in this research. Future research can include these methods for evaluation.

APPENDIX A

Section 1 Fortran Code for Welch's Method

```
program welch
implicit none

! Purpose: Welch's procedure for warm-up length determination


!Input files
!File1: Parts.txt Store on C: drive of computer. This file contains the
!number of parts produced per hour. Number of rows in this file equal the
!run length m
!
!Output files
!File2: Output Excel file. Use the data in this file to plot the graph
!and decide the warm-up length using judgment.
!
!Input variables
!m - run length in unit time
!n - number of replications
!w - window size

!Output variables


!Variable Definition
INTEGER:: m !run length
INTEGER:: n !-number of replications
INTEGER:: w!window size
INTEGER:: i !loop index
INTEGER:: j!loop index
INTEGER:: s ! temporary variable
REAL, DIMENSION(5000,5000)::!value-stores the raw data in rows and
                            !columns
REAL, DIMENSION(5000):: num !-stores the value of time associated with
                            !the raw data
REAL, DIMENSION(5000):: yibar !-stores the ensemble averages
REAL, DIMENSION(5000):: yiw !- stores the cumulative averages

!Prompt user to enter input values
WRITE(UNIT=*,FMT=*)" Enter the number of replications "
READ(UNIT=*,FMT=*)n
WRITE(UNIT=*,FMT=*)" Enter the run length "
READ(UNIT=*,FMT=*)m
!
!open file
!
OPEN(UNIT=1,FILE="C:\parts.txt",STATUS="old",ACTION="read")
OPEN(UNIT=2,FILE="C:\writeparts.xls",STATUS="replace",ACTION="write")

 !read the values from file
      do i=1,n
```

```
                do j = 1,m
                  READ(UNIT=1,FMT=*)num(j),value(j,i)
                end do
        end do

!to calculate yibar (over replications)

        do i =1,m
          yibar(i)=0
               do j=1,n
                     yibar(i)=yibar(i)+value(i,j)
               end do
          yibar(i)=yibar(i)/n
        end do

 WRITE(*,*)" yibars are calculated, now enter the value of w "
 READ(*,*)w
!
!for 1 -> w
        do i=1,w
            yiw(i)=0
               do s=1-i,i-1,1
                     yiw(i)= yiw(i)+yibar(i+s)
               end do
            yiw(i)=yiw(i)/(2*i - 1)
        end do

        do i=w+1,m-w,1
             yiw(i)=0
                do s = -w,w,1
                    yiw(i)=yiw(i)+ yibar(i+s)
                end do
              yiw(i)=yiw(i)/(2*w + 1)
        end do

         do i = 1, m
             WRITE(2,*)yiw(i)
         end do

end program welch
```

Section 2 Visual Basic for Application Code for Randomization Test used with Microsoft Excel

```
'Author: Prasad Mahajan
'Title: Randomization tests by Dr. Enver Yucesan
' Date Started: 23 rd December 2003
' Date Finished: 25th December 2003

'Purpose:  To implement the randomization test and determine the warm up length

'List of subroutines
'Batchmean()calculates the batch means
'clear()deletes previous data
'private Sub CommandButton1_Click() initiates the code to find batch means
'randomization() begins the randomization code
'sort() used to arrange the data in groups

'Input variables
```

```
'm run length; enter run length in column "runlen"
'k batch size; enter batch size in column "batchlen"
'num unit time ; enter time in column "num"
'value1 raw data; enter raw data associated with corresponding in
'column "Yi" next to "num"
'NS - read from column "NSfacto"
'Range("autocorr") should have the Excel correlation function to find the lag 1
'autocorrelation between the batch means


'Output variables
'yibar(p) ensemble averages; displayed in column "ensemble"
'mean temporarily stores the batch mean; displayed in column "yibar"
'count stores the batch means number; displayed in column "num1"
'count2 stores the iteration number for randomization test
'pvalue stores the pvalue; displayed in column "pvalue"
'NS - displays value of NS; displayed in column "NS"
'ind4 - displays warm-up length; displayed in column "WLEN1"

'Subroutine to calculate batch means


Sub Batchmean()

'Variable definitions

Dim k As Integer                    'batch size
Dim i  As Integer                   'loop index
Dim j  As Integer                   'loop index
Dim y As Integer                    'temperory variable
Dim sv As Integer                   'starting value of for loop
Dim count As Integer                'counts the number of iterations
Dim ev As Integer                   'ending value of for loop
Dim x As Integer                    'counter variable
Dim  l As Integer                   'loop index
Dim  p As Integer                   'loop index
Dim  q As Integer                   'loop index
Dim m As Integer                    'run length
Dim num(10000) As Integer           'stores the time associated with the raw data
Dim yibar(8000) As Double           'stores the batch means
Dim mean As Double                  'temperory variable used to calculate batch
                                    'means
Dim value(10000)                    'stores the batch means
Dim value1(10000, 10) As Double     'stores raw data
Dim sum1 As Double                  'used to accumulate the sum of raw data
Dim data As Integer                 'number of batches
Dim stval As Integer                'temperory variable

'read batch length,  run length and rep length
y = 0
m = Range("runlen").Offset(0, 0)
k = Range("batchlen").Offset(0, 0)
data = m / k

'Clear initial data
clear

' read data in 2 -d array
   For p = 1 To 1
       For q = 1 To m

           num(q) = Range("num").Offset(q, 0)
           value1(q, p) = Range("Yi").Offset(y + q, 0)
```

```
        Next q
        y = y + m
    Next p

' to calculate yibars (ensemble averages)
'yibar - array of ensemble averages

     For p = 1 To m
         yibar(p) = 0
             For q = 1 To 1
                 yibar(p) = yibar(p) + value1(p, q)
              Next q
         yibar(p) = yibar(p) / 1
          value(p) = yibar(p)
         Range("ensemble").Offset(p, 0) = yibar(p)
     Next p

i = 0
j = 0

' start batching
count = 1
l = 0
    For count = 1 To m / k
        l = l + k
        ev = l
        sv = l - k + 1
        suml = 0
            For x = sv To ev
            suml = suml + value(x)
            Next x
        ' calculate batch means
        mean = suml / k

        ' display results
        Range("num1").Offset(count, 0) = count
        Range("yibar").Offset(count, 0) = mean
    Next count

End Sub

private Sub CommandButton1_Click()

Dim i As Integer      'loop index
Dim m As Integer      'run length
Dim b As Integer      'number of batches
Dim k As Integer      ' batchsize
Dim data As Integer   ' number of batches


m = Range("runlen")
Range("Batchlen") = 1
k = Range("batchlen")

    For i = 1 To 100

           If (i <> 1) Then
               Range("Batchlen") = Range("Batchlen") * 2
               k = k * 2
          End If
         data = Range("runlen") / Range("batchlen")
         b = m / k
        'batchmean check'''''''
         Batchmean

         ' Check for autocorrelation
              If Abs(Range("autocorr")) < 0.5 And data < 2 Then
                  MsgBox (" Autocorrelation not present batch size too long ")
                  GoTo finish:
                      Else

                      If Abs(Range("autocorr")) < 0.5 And (data >= 2) Then
                           MsgBox (" Autocorrelation not present ")
                           GoTo finish:

                      End If
              End If
    Next i
finish:
End Sub

Private Sub CommandButton2_Click()
       clear
End Sub

Sub clear()
    ActiveSheet.Range("C18:E7118") = ""
    ActiveSheet.Range("F13") = ""
    ActiveSheet.Range("I13:J13") = ""
    ActiveSheet.Range("F38:N7444") = ""
    ActiveSheet.Range("I35") = ""
End Sub
```

Subroutine to calculate randomization tests

```
Sub randomization ()

     Dim i As Integer                          'loop index
     Dim j As Integer                          'loop index
     Dim NG1 As Integer                        'number of elements in group 1
     Dim  NG2 As Integer                       'number of elements in group 2
     Dim m As Integer                          'run length
     Dim  b As Integer                         'number of batches
     Dim ix As Integer                         'loop index
     Dim l(1000) As Integer                    'stores the limit for sorting
     Dim ind3 As Integer                       'temperory variable
     Dim ind4 As Integer                       'temperory variable
     Dim A(1000) As Integer                    'stores the numbers for sorting
                                               ' data in  groups
     Dim M1(1000) As Double                    'Array to find the mean of group
                                               '1
     Dim W(1000) As Double                     'Array to find the mean of group
                                               '2
     Dim count3 As Double                      'counter variable
     Dim count2 As Double                      'counter variable
     Dim sumng1 As Double                      'stores sum for group 1
     Dim sumng2 As Double                      'stores sum for group 2
     Dim yibar(1000) As Double                 'stores batch means
     Dim f as Double                           'stores the f-statistic
     Dim  nge As Double                        'stores the nge value
     Dim pvalue As Double                      'stores the p-value
     Dim NS As Double                          'stores the value of NS
```

```
'Read input data

b = Range("Batches")
m = Range("Runlen")
count3 = 0
    For i = 1 To b
        yibar(i) = Range("Yibar").Offset(i, 0)
    Next i

'' loops through one batch at a  time

    For ix = 1 To b / 2

         sumng1 = 0
         sumng2 = 0

         'initialize the NGs to zero
         NG1 = 0
         NG2 = 0

        'Assign sizes to groups 1 and 2
         NG1 = NG1 + ix
         NG2 = b - ix

         NS = Range("NSfacto")
         Range("NS").Offset(ix, 0) = NS

         'Calculate initial mean of group1

            For j = 1 To NG1
                W(j) = yibar(j)
                sumng1 = sumng1 + W(j)
            Next j
         sumng1 = sumng1 / NG1

         'Calculate initial mean of group2

            For j = NG1 + 1 To b
                M1(j) = yibar(j)
                sumng2 = sumng2 + M1(j)
            Next j
         sumng2 = sumng2 / (b - NG1)

        'Calculate the critical test statistic for the test
        f = Abs(sumng2 - sumng1)

        'Prepare the initial array A ready to go for shuffling
            For j = 1 To NG1
                A(j) = j
            Next j

        'Send array Array for shuffling
         nge = 0
            Call sort(NG1, NG2, b, yibar, f, nge, A, count2, count1)
         nge = nge - 1
        'Calculate p -value for the test
         pvalue = (nge + 1) / (NS + 1)
         Range("pvalue").Offset(ix, 0) = pvalue
         MsgBox " P value is " & pvalue
            If (pvalue > 0.05) Then
                 ind3 = 1
                 ind4 = ix
                 GoTo finish:
```

```
           Else
              ind3 = 2
           End If

count3 = count3 + count2
   Next ix

finish:

 'Determine if warm -up length has been found or you need more data
     If (ind3 = 1) Then
        MsgBox (" Warm-up length found ")
        Range("WLEN1") = ind4
     End If

      If (ind3 = 2) Then
        MsgBox (" Warm up length not found ")
        MsgBox (" Increase the run length and run me again ")
      End If

 ' END OF THE TEST

End Sub

   Sub sort(NG1, NG2, b, yibar, f, nge, A, count2, count1)

      Dim sumng1new As Double      'stores the mean for arranged group 1
      Dim sumng2new As Double      'stores the mean for arranged group 2
      Dim fpseudo As Double        ' stores the pseudo statistic
      Dim j As Integer             ' loop index
      Dim in1 As Integer           ' indicator variable
      Dim Ar(10000) As Integer     'stores the array of numbers used for
                                   'arranging data in groups
      Dim N As  Integer            'stores the limit of iteration
      Dim I1 As Integer            'loop index
      Dim ind1 As Integer          'temporary variable
      Dim  i2 As Integer           'indicator variable
      Dim l(10000)As integer       ' stores the limit for iteration
      Dim cali As Integer          ' temporary variable

cali = 2
For j = 1 To NG1
l(j) = b - NG1 + j
Next j

For I1 = 1 To b
Ar(I1) = A(I1)
Next I1

' Initialize variables
count1 = 0
N = NG1
nge = 0
sumng1new = 0
sumng2new = 0

'   !Loop will terminate when all shuffling is over
Do  ' loop 0

   ' loop to shuffle the numbers and calculate means
       Do ' WHILE loop 1
1:
                count1 = count1 + 1
                count2 = count2 + 1
                Range("count2") = count1

                'To calculate mean of group 1
                 sumng1new = 0

                    For I1 = 1 To NG1
                                sumng1new = sumng1new + yibar(Ar(I1))
                    Next I1
                  sumng1new = sumng1new / NG1
                  sumng2new = 0

                    For i2 = 1 To b
2:
                                 ind1 = 0
                                       For j = 1 To NG1
                                              If (i2 = Ar(j)) Then
                                                     ind1 = 1
                                                     GoTo out:
                                              End If
                                       Next j
out:
                                       If (ind1 = 1) Then
                                               i2 = i2 + 1
                                              GoTo 2:
                                       End If

                                sumng2new = sumng2new + yibar(i2)
                    Next i2
          sumng2new = sumng2new / NG2

          'Calculate the pseudo statistic
                  fpseudo = Abs(sumng1new - sumng2new)
                  'Range("pseudo").Offset(count2, 0) = fpseudo
                  'increment nge
                         If (fpseudo >= f) Then
                                nge = nge + 1
                     End If
               '!if last element in  shuffle array is less than its limit the
               '! increment it by 1
                   If ((Ar(N)) < l(N)) Then
                         Ar(N) = Ar(N) + 1
                         GoTo 1:
                   End If
               'if limit has been reached go back one element
               ' you have to exit this loop first
                   If (Ar(N) = l(N)) Then
                         GoTo outof1: ' Exit WHILE LOOP1 AND ARRIVE AT **
                   End If

              Loop Until count1 > 20000000 'loop 1

outof1:
      'if the limit of first element has reached that means you are done
      ' come out of this loop too. Remember we are starting with the last
element
             If (Ar(1) = l(1)) Then
                  GoTo outof0:
             End If
      'decrement N by 1 if limit has reached (go back 1 step)
      N = N - 1
      I1 = N
      ' increment proceeding elements if they are within limit
```

```
          cali = 2

             '! WHILE loop 2(To increment all proceeding elements if within
             'limit)

                  Do
                        If (Ar(I1) < l(I1)) Then
                            Ar(I1) = Ar(I1) + 1
                            in1 = 0

                             For j = I1 + 1 To NG1
                                 in1 = in1 + 1
                                  If ((Ar(I1) + in1) < l(j)) Then
                                       Ar(j) = Ar(I1) + in1
                                       N = j
                                  Else
                                       N = j - 1
         '                             cali = 1
                                       GoTo outofW2:
                                  End If
                             Next j

                            cali = 1
                            GoTo outofW2: ' Exit this loop
                       Else
                            I1 = N - 1
                            'if the preceding element has reached limit go
                            'back by 1 element
                            cali = 1
                            GoTo outofW2:
                            'check for their proceding elements
                       End If

                  Loop Until cali = 2 'WHILE 2
    outofW2:

    Loop Until count1 > 20000000 'loop 0

outof0:
fini:
End Sub

Private Sub CommandButton3_Click()
randomization
End Sub
```

Section 3 Visual Basic for Application Code for Conway Rule used with Microsoft Excel

Note: Paste this part of code in sheet 1 module

```
Private Sub CommandButton1_Click()
    'Name: Prasad Mahajan
    'Date: 12/15/2003

    'Input Variables:
    'm run length
    'n number of replications
    'value raw data

    'Output Variables
    'purpose: To program conway rule
            'Here: To arrange data in rows and columns


    Dim i As Integer                 'loop index
    Dim j As Integer                 'loop index
    Dim k As Integer                 'position specifier
    Dim m As Integer                 'run length
    Dim n As Integer                 'number of replications
    Dim value(10000,10) As Double 'raw data

    'read data
    k = 5
    n = Worksheets("sheet1").Cells(3, 1)
    m = Worksheets("sheet1").Cells(3, 2)
    '
    'loops across the replications
      For j = 3 To n + 2
        'loops thru the run length
         For i = 6 To m + 5
           k = k + 1
           Worksheets("sheet1").Cells(i, j) = Worksheets("sheet1").Cells(k, 2)
           'store data in array value
           value(i - 5, j - 2) = Worksheets("sheet1").Cells(i, j)
         Next i
      Next j
End Sub
```

Note: Paste this part of code in sheet 2 module

```
Private Sub CommandButton1_Click()
    'Name: Prasad Mahajan
    'Date: 12/15/2003
    'purpose: To program conway rule
    '         HERE: To find the min and max of remaining data for each point in
              'each replication
' variable declaration

    Dim i As Integer                 'loop index
    Dim j As Integer                 'loop index
    Dim k As Integer                 'position specifier
    Dim m As Integer                 'run length
    Dim n As Integer                 'number of replications
    Dim l As Integer                 'temperory variable
    Dim value(10000,10) As Double 'raw data
    Dim maxi As Double               'stores the maximum value
    Dim mini As Double               'stores the minimum value

    n = Worksheets("sheet1").Cells(3, 1)
    m = Worksheets("sheet1").Cells(3, 2)

      ' read data from sheet 1
      For j = 3 To n + 2
         For i = 6 To m + 5
           k = k + 1
           value(i - 5, j - 2) = Worksheets("sheet1").Cells(i, j)
         Next i
      Next j

    ' initialize variables
    l = 0
      For j = 1 To n
        l = l + 2
        'calculate max and mins for all replications
            For i = 1 To m
              'set maxi and mini to very low and very high values
              maxi = -10
              mini = 10000
                  For k = i To m
                      If (value(k, j) > maxi) Then
                        maxi = value(k, j)
                      End If
                      If (value(k, j) < mini) Then
                        mini = value(k, j)
                      End If
                  Next k
              ' display maxi and mini of remaining data for each data point in
              'each replication
              Worksheets("sheet2").Cells(i + 4, 1) = maxi
              Worksheets("sheet2").Cells(i + 4, 1 + 1) = mini
            Next i
      Next j
End Sub
```

Note: Paste this part of code in sheet 3 module

```
'Name: Prasad Mahajan
'Purpose : To program Conway rule
'          Here: To find the warm up length for each replication

Private Sub CommandButton1_Click()
'
    ' Variable declaration
    Dim i As Integer 'loop index
    Dim j As Integer 'loop index
    Dim n As Integer 'number of replications
    Dim m As Integer 'run length
    Dim k As Integer 'batch size
    Dim in1 As Integer      'indicator variable

    n = Worksheets("sheet1").Cells(3, 1)
    m = Worksheets("sheet1").Cells(3, 2)
    k = 0
        'loop to go across replications
        For j = 1 To n
            k = k + 2
            in1 = 0
              'loop through the run length
              For i = 1 To m
                  ' if the number is neither the max nor the min in the remaining
                  'series, then that is the truncation point
                    If ((Worksheets("sheet1").Cells(i + 5, j + 2) <
                    Worksheets("sheet2").Cells(i + 4, k)) And
                    (Worksheets("sheet1").Cells(i + 5, j + 2) >
                    Worksheets("sheet2").Cells(i + 4, k + 1))) Then
                         in1 = 1
                         Worksheets("sheet3").Cells(8, j + 3) = i
                         Exit For
                    End If
              Next i
        Next j

End Sub
```

Section 4: Visual Basic for Application code for Crossing the means rule used with Microsoft Excel

```
Private Sub CommandButton1_Click()
'
   'Purpose: To model the crossing of means rule
   'Name: Prasad Mahajan

'Input Variables
'm run length
'n0 number of crossings desired
'column "data" should contain raw data

'Output Variables
'sumzz number of times the mean is crossed; displayed in column "wj2"
'warmuplen displays the warm up length; displayed in column warmup
'Column "cumulative" displays the cumulative mean


'Declare variables

Dim i As Integer                   'loop index
Dim m As Integer                   'run length
Dim in1As Integer                  'indicator variable
Dim i1 As Integer                  'loop index
Dim wj(20000)As Integer            'counts the number of times mean is crossed
Dim warmuplen As Integer           'stores the warm-up length
Dim no As Integer                  'number of crossings
Dim j As Integer                   'loop index
Dim xx As Integer                  'loop index
Dim X1 As Single                   'consecutive data point with X2
Dim X2 As Single                   'consecutive data point with X1
Dim mean As Single                 'cumulative mean of raw data
Dim sum As Single                  'stores the cumulative sum
Dim sumzz As Single                'sums up the number of times mean crosses

   in1 = 0
   'Initialize variables

   ' Read input values
   m = Range("runlen").Offset(1, 0)
   no = Range("crosses").Offset(1, 0)
   sum = 0
   mean = 0
   '
   'Calculate original mean
   sum = 0
      For i = 1 To m
        sum = sum + Range("Data").Offset(i, 0)
      Next i
   sum = sum / m
   Range("actual") = sum
   'calculate cumulative means
      For i = 1 To m
        sum = 0
          For j = 1 To i
             sum = sum + Range("data").Offset(j, 0)
          Next j
        Range("cumulative").Offset(i, 0) = sum / i
        sum = sum / i
          For xx = 1 To m
              wj(xx) = 0
          Next xx
        '
        ' loop for finding the number of times mean is crossed

          For i1 = 1 To i
              X1 = Range("cumulative").Offset(i1, 0)
              X2 = Range("cumulative").Offset(i1 + 1, 0)
              ' check if mean is crossed or not

                If ((X1 > sum And X2 < sum) Or (X1 < sum And X2 > sum)) Then
                  ' increment wj if yes
                  wj(i) = wj(i) + 1
                  ' if wj equals the prespecified number of crossings then steady
                  `state has reached

                        If (wj(i) = no) Then
                            warmuplen = i
                            Range("warmup").Offset(1, 0) = warmuplen
                            in1 = 11
                            GoTo finish1
                        End If
                Else
                End If
          Next i1
    finish1:
        sumzz = 0

          For xx = 1 To m
              sumzz = sumzz + wj(xx)
          Next xx

        Range("wj2").Offset(i, 0) = sumzz

          If (in1 = 11) Then GoTo finish:
      Next i
    finish:

End Sub
```

APPENDIX B

Section 1 K-S Test for Normal Distribution

This version of the test is developed by H.W. Lilliefors (1967) [28]. The test is based on a comparison between the cumulative distributions of the empirical and normal distribution.
The hypotheses are:

H_0: The failure times are normal
H_1: The failure times are not normal

The test statistic is $D_n = \max\{D_1, D_2)$ where-

$$D_1 = \max_{1 \le i \le m} \left\{ \Phi(\frac{Y_i - \bar{Y}}{s}) - \frac{i-1}{m} \right\}$$

$$D_2 = \max_{1 \le i \le n} \left\{ \frac{i}{m} - \Phi(\frac{Yi - \bar{Y}}{s}) \right\}$$

$$\bar{Y} = \sum_{i=1}^{m} \frac{Y_i}{m}$$

$$s^2 = \frac{\sum_{i=1}^{m} (Y_i - \bar{Y})^2}{m-1}$$

If $D_m < D_{crit}$, then accept H_0; otherwise accept H_1. The value of D_{crit} can be found in the table in Appendix D.

Section 2 Anderson Darling Test for Testing Normality

This test is explained in context of the modified SPC method as given by Stewart Robinson. The test is applied to last half of the batch means data.
Let
b = number of batches
sl = int ((b/2) + 0.5) + 1
$n = (b - sl)$
Initially there will be b batch means. Retain only the batch means from batches sl to b.

$\overline{Y}_{mi}$ = batch means for $i = sl$ to b

$\overline{Y}_{mis}$ = $\overline{Y}_i$'s s sorted in ascending order

$$\text{AD} = \left\{-n - \sum_{i=1}^{n}(((2\,x\,i)-1)/n) \text{ x } \left[\log(\phi(\overline{Y}_{is}) + \log(1-\phi(\overline{Y}_{is})\right]\right\} \text{ x } \left[(1+\frac{4}{n}) - (\frac{25}{n^2})\right]$$

Significance level	Critical Value
10 %	0.632
5%	0.75
2.5%	0.82
1%	1.029

Null hypothesis that the data is normal is rejected if AD < critical value for a given level of significance.

Section 3 Von Neumann's Test for Autocorrelation

This test is modified and is given in context with the modified SPC method.

m = run length

k = batch size

b = number of batches

sl = int(b/2+0.5) + 1

$$\text{mean} = \frac{\sum_{i=((sl-1)xk)+1}^{m} \overline{Y}_i}{n}$$

$n = m - (((sl - 1) \text{ x } k) + 1)$

$$\text{Von Neumann's test statistic} = \frac{\sum_{i=sl}^{b}\overline{Y}_{mi} - M)(\overline{Y}_{mi+1} - M)}{\sum_{i=sl}^{b}\overline{Y}_{mi} - M} + \frac{(\overline{Y}_{msl} - M)^2 + (\overline{Y}_{mb} - M)^2}{2\sum_{i=sl}^{b}\overline{Y}_{mi} - M}$$

The hypothesis of no autocorrelation is rejected if the value of test statistic is > 1.9

APPENDIX C

Results for Approximate Randomization Test

Following are the results for Randomization Test when the changes suggested in section 5.1 are incorporated.

Run length = 1000 hours, NS =1999							
Model	Final Mean	Average Run length (hours)	Final MSE	Final Var	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Original	22.747	1000	0.060	0.0212	1220	-7.73	-0.14
Modified	22.747	1000	0.060	0.0212	14.4	-7.78	-0.28
Run length = 5200 hours, NS =1999							
Model	Final Mean	Average Run length (hours)	Final MSE	Final Var	Average Computing Time (seconds)	% Change in MSE	%Change in variance
Original	22.6201	1000	0.060	0.019	750	-0.24	-7.21
Modified	22.622	1000	0.0663	0.0188	17	-9.54	-0.42

Table C.1: Results for Randomization Test on Type III Model for Run Lengths 1000 Hours and 5200 Hours

APPENDIX D

Siman Code for Experimental Model

```
;
;
;     Model statements for module:  Create 1
;

17$            CREATE,         1,MinutesToBaseTime(0.0),Entity
1:MinutesToBaseTime(EXPO( 4.5 )):NEXT(18$);

18$            ASSIGN:         Create 1.NumberOut=Create 1.NumberOut +
1:NEXT(0$);


;
;
;     Model statements for module:  Decide 1
;
0$             BRANCH,         1:
                               With,50/100,1$,Yes:
                               With,30/100,5$,Yes:
                               Else,9$,Yes;

;
;
;     Model statements for module:  Process 15
;
9$             ASSIGN:         T3 to C2.NumberIn=T3 to C2.NumberIn + 1:
                               T3 to C2.WIP=T3 to C2.WIP+1;
26$            QUEUE,          T3 to C2.Queue;
25$            SEIZE,          2,VA:
                               Cell C2,1:NEXT(24$);

24$            DELAY:          EXPO( 0.9 ),,VA;
23$            RELEASE:        Cell C2,1;
71$            ASSIGN:         T3 to C2.NumberOut=T3 to C2.NumberOut + 1:
                               T3 to C2.WIP=T3 to C2.WIP-1:NEXT(10$);


;
;
;     Model statements for module:  Process 16
;
10$            ASSIGN:         T3 to C1.NumberIn=T3 to C1.NumberIn + 1:
                               T3 to C1.WIP=T3 to C1.WIP+1;
77$            QUEUE,          T3 to C1.Queue;
76$            SEIZE,          2,VA:
```

```
                                   Cell C1,1:NEXT(75$);

75$          DELAY:                EXPO( 0.5 ),,VA;
74$          RELEASE:              Cell C1,1;
122$         ASSIGN:               T3 to C1.NumberOut=T3 to C1.NumberOut + 1:
                                   T3 to C1.WIP=T3 to C1.WIP-1:NEXT(11$);


;
;
;     Model statements for module:  Process 17
;
11$          ASSIGN:               T3 to C4.NumberIn=T3 to C4.NumberIn + 1:
                                   T3 to C4.WIP=T3 to C4.WIP+1;
128$         QUEUE,                T3 to C4.Queue;
127$         SEIZE,                2,VA:
                                   Cell C4,1:NEXT(126$);

126$         DELAY:                EXPO(0.7 ),,VA;
125$         RELEASE:              Cell C4,1;
173$         ASSIGN:               T3 to C4.NumberOut=T3 to C4.NumberOut + 1:
                                   T3 to C4.WIP=T3 to C4.WIP-1:NEXT(12$);


;
;
;     Model statements for module:  Process 18
;
12$          ASSIGN:               T3 to C5.NumberIn=T3 to C5.NumberIn + 1:
                                   T3 to C5.WIP=T3 to C5.WIP+1;
179$         QUEUE,                T3 to C5.Queue;
178$         SEIZE,                2,VA:
                                   Cell C5,1:NEXT(177$);

177$         DELAY:                EXPO( 0.1 ),,VA;
176$         RELEASE:              Cell C5,1;
224$         ASSIGN:               T3 to C5.NumberOut=T3 to C5.NumberOut + 1:
                                   T3 to C5.WIP=T3 to C5.WIP-1:NEXT(13$);


;
;
;     Model statements for module:  Process 19
;
13$          ASSIGN:               T3 to C3.NumberIn=T3 to C3.NumberIn + 1:
                                   T3 to C3.WIP=T3 to C3.WIP+1;
230$         QUEUE,                T3 to C3.Queue;
229$         SEIZE,                2,VA:
                                   Cell C3,1:NEXT(228$);

228$         DELAY:                EXPO( 1.4 ),,VA;
227$         RELEASE:              Cell C3,1;
275$         ASSIGN:               T3 to C3.NumberOut=T3 to C3.NumberOut + 1:
                                   T3 to C3.WIP=T3 to C3.WIP-1:NEXT(6$);


;
```

```
;
;     Model statements for module:  Dispose 1
;
6$            ASSIGN:        Dispose 1.NumberOut=Dispose 1.NumberOut +
1;
278$          DISPOSE:       Yes;


;
;
;     Model statements for module:  Process 1
;
1$            ASSIGN:        C3.NumberIn=C3.NumberIn + 1:
                             C3.WIP=C3.WIP+1;
282$          QUEUE,         C3.Queue;
281$          SEIZE,         2,VA:
                             Cell C3,1:NEXT(280$);

280$          DELAY:         EXPO( 0.4 ),,VA;
279$          RELEASE:       Cell C3,1;
327$          ASSIGN:        C3.NumberOut=C3.NumberOut + 1:
                             C3.WIP=C3.WIP-1:NEXT(2$);


;
;
;     Model statements for module:  Process 2
;
2$            ASSIGN:        C2.NumberIn=C2.NumberIn + 1:
                             C2.WIP=C2.WIP+1;
333$          QUEUE,         C2.Queue;
332$          SEIZE,         2,VA:
                             Cell C2,1:NEXT(331$);

331$          DELAY:         EXPO( 0.5 ),,VA;
330$          RELEASE:       Cell C2,1;
378$          ASSIGN:        C2.NumberOut=C2.NumberOut + 1:
                             C2.WIP=C2.WIP-1:NEXT(3$);


;
;
;     Model statements for module:  Process 3
;
3$            ASSIGN:        C1.NumberIn=C1.NumberIn + 1:
                             C1.WIP=C1.WIP+1;
384$          QUEUE,         C1.Queue;
383$          SEIZE,         2,VA:
                             Cell C1,1:NEXT(382$);

382$          DELAY:         EXPO( 0.6 ),,VA;
381$          RELEASE:       Cell C1,1;
429$          ASSIGN:        C1.NumberOut=C1.NumberOut + 1:
                             C1.WIP=C1.WIP-1:NEXT(4$);


;
```

```
;
;      Model statements for module:  Process 4
;
4$             ASSIGN:         C5.NumberIn=C5.NumberIn + 1:
                               C5.WIP=C5.WIP+1;
435$           QUEUE,          C5.Queue;
434$           SEIZE,          2,VA:
                               Cell C5,1:NEXT(433$);

433$           DELAY:          EXPO( 1.2 ),,VA;
432$           RELEASE:        Cell C5,1;
480$           ASSIGN:         C5.NumberOut=C5.NumberOut + 1:
                               C5.WIP=C5.WIP-1:NEXT(6$);


;
;
;      Model statements for module:  Process 5
;
5$             ASSIGN:         C4.NumberIn=C4.NumberIn + 1:
                               C4.WIP=C4.WIP+1;
486$           QUEUE,          C4.Queue;
485$           SEIZE,          2,VA:
                               Cell C4,1:NEXT(484$);

484$           DELAY:          EXPO(1.2 ),,VA;
483$           RELEASE:        Cell C4,1;
531$           ASSIGN:         C4.NumberOut=C4.NumberOut + 1:
                               C4.WIP=C4.WIP-1:NEXT(7$);


;
;
;      Model statements for module:  Process 13
;
7$             ASSIGN:         T2 to C1.NumberIn=T2 to C1.NumberIn + 1:
                               T2 to C1.WIP=T2 to C1.WIP+1;
537$           QUEUE,          T2 to C1.Queue;
536$           SEIZE,          2,VA:
                               Cell C1,1:NEXT(535$);

535$           DELAY:          EXPO( 0.8 ),,VA;
534$           RELEASE:        Cell C1,1;
582$           ASSIGN:         T2 to C1.NumberOut=T2 to C1.NumberOut + 1:
                               T2 to C1.WIP=T2 to C1.WIP-1:NEXT(8$);


;
;
;      Model statements for module:  Process 14
;
8$             ASSIGN:         T2 to C3.NumberIn=T2 to C3.NumberIn + 1:
                               T2 to C3.WIP=T2 to C3.WIP+1;
588$           QUEUE,          T2 to C3.Queue;
587$           SEIZE,          2,VA:
                               Cell C3,1:NEXT(586$);
```

```
586$            DELAY:          EXPO( 1 ),,VA;
585$            RELEASE:        Cell C3,1;
633$            ASSIGN:         T2 to C3.NumberOut=T2 to C3.NumberOut + 1:
                                T2 to C3.WIP=T2 to C3.WIP-1:NEXT(6$);


;
;
;     Model statements for module:  Create 2
;

636$            CREATE,
1,MinutesToBaseTime(0.0),new:MinutesToBaseTime(0),1:NEXT(637$);

637$            ASSIGN:         Flagman.NumberOut=Flagman.NumberOut +
1:NEXT(14$);


;
;
;     Model statements for module:  Delay 1
;
14$             DELAY:          1,,Other:NEXT(15$);


;
;
;     Model statements for module:  ReadWrite 1
;
15$             WRITE,          File 1,"(I5,3X,F6.3)":
                                BaTCH,
                                EntitiesWIP(Entity 1):NEXT(16$);


;
;
;     Model statements for module:  Assign 1
;
16$             ASSIGN:         BaTCH=bAtch+1:NEXT(14$);


;
;
;     Model statements for module:  Variable 2
;
```

REFERENCES

[1] Amemiya, T. *"Generalized Least Squares with an Estimated Autocovariance Matrix"* Econometricia, 41, 723-732 (1973).

[2] Banks, J., J.S. Carson, and B.L. Nelson 1996. *Discrete Event System Simulation* 2nd ed. Upper Saddle River, NJ: Prentice-Hall.

[3] Billingsley, P., *Convergence of probability Measures*, Wiley , New York, 1968

[4] Bissel, D 1994 *"Statistical methods for SPC and TQM"* London: Chapman and Hall.

[5] Blomqvist, N. 1970 *On the transient Behavior of the GI/G/1 Waiting-times.* Skand Aktuarietidskr., 118-129.

[6] Cash, C.R. Nelson, B. L., Long, J. M., Dippold, D. G., and Pollard W.P., *"Evaluation of Tests for initial-condition bias"* Proceedings of the 1992 Winter simulation conference 1992, pp 577-585.

[7] Charles Ebeling *"An Introduction to Reliability and Maintainability Engineering"* Tata McGraw Hill Publication New Delhi 1997.

[8] Christos Alexopoulos and Andrew F. Seila *"Advanced methods for simulation output analysis"* Proceedings of the 1998 Winter Simulation conference, D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannam, eds.

[9] Conway R.W. *" Some Tactical Problems in Digital Simulation"* Management Science, Volume 10, Issue 1, (Oct., 1963), 47-61.

[10] David Goldsman, Gamze Tokol, "*Output Analysis Procedures for Computer Simulations" Proceeding of the 2000 Winter Simulation Conference* J.A. Joines, R. R. Barton, K. Kan, and P.A. Fishwick, eds.

[11] Fishman G.S. 1971. "*Estimating sample size in computer simulation experiments*". Management Science 18:21-37.

[12] Fishman G.S. 1972 "*Bias Considerations in Simulation Experiments*" Operations Research, Volume 20, Issue 4 (Jul. – Aug., 1972), 785-790.

[13] Fishman G.S. 1973 "*Concepts and Methods in discrete Event Digital Simulation*". New York Wiley.

[14] Fishman G.S. 1996 " *Monte Carlo Concepts, Algorithms, and Applications*" Springer Verlag New York.

[15] Gafarian, A.V., C.J Ancker, and T. Morisaku. 1978. "*Evaluation of commonly used rules for detecting steady-state in computer simulation*". Naval Research Logistics Quarterly 25: 511-529.

[16] Goldsman D., L.W. Schruben and J. J. Swain "*Tests for transient means in simulation time series*" volume 41, 1994, Pages 171-187 Naval Research Logistics, New York Wiley.

[17] Heidelberger. P and Welch P.D. "*A spectral method for confidence interval generation and run length control in simulations.*" ACM 24, 233-245, 1981

[18] Heidelberger. P., and Welch P. D. "*Simulation run length control in the presence of an initial transient*" Operations research volume 31, No. 6, November – December 1983.

[19] Jackway P.T. and B.S. DeSilva, 1992. "A methodology for initialization bias reduction in *computer simulation output*". Asia Pacific Journal of Operations Research 9: 87-100.

[20] Kelton, W. D., *"The Startup Problem in Discrete Event Simulation"* Technical report No. 80-1, Department of Industrial Engineering, University of Wisconsin, 1980.

[21] Kelton W.D. and A.M. Law 1983. *"A new approach for dealing with the startup problem in discrete event simulation"*. Naval Research Logistic Quarterly 30: 641-658.

[22] Law, A.M. 1975. *A Comparison of Two Techniques for Determining the Accuracy of Simulation Output,* Technical report 75-11, Dept of Industrial Engineering, University of Wisconsin Madison.

[23] Law A. M. 1977. *Confidence Intervals in Discrete Event Simulation: A Comparison of Replication and Batch Means.* Naval Res. Logistics Quarterly. 24, 667-678.

[24] Law A. M Statistical Analysis of Simulation Output Data Operations Research 31, Issue 6, Simulation (Nov – Dec., 1983), 983-1029.

[25] Law A. M. And W. D. Kelton. 1979 *Confidence intervals for Steady State Simulations; I.A Survey of Fixed Sample Size Procedures*, Technical Report 78-, Dept of Industrial Engineering, University of Wisconsin Madison.

[26] Law, A.M., AND W. D. Kelton. 1982 *Simulation Modeling and Analysis.* Mc Graw-Hill publication New York.

[27] Law, A.M., and W.D. Kelton. 2000. *"Simulation Modeling and Analysis"*. 3rd Ed. New York: McGraw Hill.

[28] Lilliefors, H.W. *"On the Kolmogorov – Smirnov Test for Normality with Mean and Variance Unknown"* Journal of American Statistical Association, Vol. 62 (1967), .pp. 399-402.

[29] Nelson B.L. Handbook of Industrial Engineers, second edition, Wiley NewYork, Chapter 102 Page 2567, *"Statistical analysis of simulation results"*

[30] Noreen Eric W. *"Computer Intensive Methods for Testing Hypotheses An Introduction"* 1989, New York.

[31] Pawlikowski K. 1990. *"Steady State simulation of queuing processes: a survey of problems and solutions"*. Computing Surveys 22: 123-170.

[32] Robinson Stewart *"A Statistical Process Control Approach For Estimating the Warm-up Period"* Proceedings of the 2002 Winter Simulation Conference E Yücesan, C.-H. Chen, J. L. Snowdon, and J.M.Charnes, Eds.

[33] Robinson Stewart 2004 *"A Statistical Process Control Approach to select a warm-up period for discrete event simulation"* Unpublished paper currently in review

[34] Schruben, L.W. 1982 *"Detecting initialization bias in simulation output"* Operations Research 30 No 3. : 569-590.

[35] Schruben L., Singh H., Tierney L., *"Optimal tests for initialization bias in simulation output"*. Operations Research volume 31, No. 6, November-December 1983.

[36] Snell, M., and L. W. Schruben 1979. *"Weighting Simulation Data to reduce Initialization Effects"*, Technical Report 395, School of Industrial Engineering, Cornell University.

[37] Spratt, S. C. 1998. *"Heuristic for startup problem"* M.S. thesis, Department of Systems Engineering, University of Virginia.

[38] Susan M. Sanchez *"ABC's of Output Analysis"* Proceedings of 1999 Winter Simulation Conference. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, Eds.

[39] Welch, P. 1983. *"The statistical analysis of simulation results."* In The computer modeling handbook, ed. S. Lavenberg, 268-328. New York: Academic Press.

[40] White K. Preston Jr. 1997 *"An effective Truncation Heuristic for Bias Reduction in* Simulation Output". Simulation 69 6 323 –334.

[41] White K. Preston Jr. , Michael J Cobb, Stephen C Spratt " *A comparison of five steady state truncation heuristics for simulation* " Proceedings of 2000 winter simulation conference J.A Joines R. P. Barton, K.Kang and P.A. Fishwick, eds.

[42] Wilson J.R. and A.B. Pritsker. 1978. *"A survey of research on the simulation startup problem"*. Simulation 31: 55-59.

[43] Wilson J.R. and A.B. Pritsker. *"Evaluation of startup Policies in Simulation Experiments"* Simulation. Volume 31 no. 3 September 1978.

[44] Yucesan E. 1993. *"Randomization tests for initialization bias in simulation output"*. Naval Research Logistics Quarterly 40: 643-663.

[45] Arena 5 Simulation Software Developed by Rockwell Software Inc.

[46] RAQS is a Queuing Software developed by the Center for Computer Integrated Manufacturing at Oklahoma State University Stillwater.

BIBLIOGRAPHY

[1] Fishman George *"Principles of Discrete event simulation* " Wiley New York 1978

[2] Gross and Harris *"Fundamentals of Queuing Theory"* 1998 John Wiley and sons Third edition.

[3] J. Johnston *"Econometric Methods"* Third edition, McGraw-Hill publication 1984.

[4] Law, A.M., AND W. D. Kelton. 2000 *Simulation Modeling and Analysis.* Third Edition Mc Graw-Hill publication New York.

[5] *"Computer Performance Modeling Handbook"* edited by Stephen S. Lavenberg.

[6] Paul Bratley, Bennett L. Fox, Linus E. Scharge *"A guide to Simulation"*.

[7] Wonnacott R., Wonnacott T. *"Econometrics"* Second edition Wiley New York

#V

VITA

Prasad Suresh Mahajan

Candidate for the Degree of

Master of Science

Thesis: Analysis of methods to detect warm-up length in steady state simulation

Major Field: Industrial Engineering

Biographical

Education: Graduated from Loyola High School, Pune, India in March 1996. Received Bachelor of Engineering degree in Mechanical Engineering from University of Pune, India. Completed the requirements for Master of Science in Industrial Engineering at Oklahoma State University, Stillwater, USA, in May 2004.

Experience: Manufacturing Engineer at Jaws Manufacturing Company, Pune, India from June 2000 to June 2001. Teaching assistant in the School of Industrial Engineering and management for compute programming course for the Spring 2002 semester. Lead teaching assistant in the School of Industrial Engineering and Management for Computer programming course from August 2002 to December 2003.

Professional Memberships: Institute for Operations Research and Management Sciences, Institute for Industrial Engineers and American Productivity and Inventory Control Society.